AMERICAN UNIVERSITY OF BEIRUT

MAROUN SEMAAN FACULTY OF ENGINEERING & ARCHITECTURE

American University of Beirut

School of Engineering and Architecture

Department of Electrical and Computer Engineering

A database design for the Dakowdas Competitive Programming Website

By

Dana Kossaybati (Group Leader)

(dak39@mail.aub.edu)

Mohammad Khaled Charaf

(mmc51@mail.aub.edu)

Mohammad Ayman Charaf

(mmc50@mail.aub.edu)

A REPORT

submitted to Dr. Hussein Bakri in partial fulfillment of the requirements of phase 4 of the database project for the course EECE433 – Database Systems

November 2023

Table of Contents

contestantUsername [PK] text	a boolean	b boolean	c boolean	d boolean	e boolean	f boolean
Khaled47	true	true	false	false	false	false
ay_charaf	true	false	false	false	true	false
cutiepie	true	true	true	false	false	false
TheJoker	true	false	false	false	false	false

1- Introduction

The exciting and dynamic world of competitive programming presents a knowledge-filled environment for creative minds with a passion for problem solving. Competitive programming has gained immense popularity over the years, attracting participants from around the globe. This popularity brings with it a substantial amount of data, including user profiles, contest results, problem sets, and discussions. Managing this data effectively is essential not only for the website's performance but also for ensuring a seamless user experience. To streamline the processes, ensure fair competition, and provide an enriching experience for programmers and coding enthusiasts, our database project for the competitive programming website **Dakowdas** aims to satisfy the needs and wants of programmers.

It’s quintessential to take into consideration all the parties involved while maintaining CIA: Confidentiality, Integrity, and Availability. As owners of the website, we’d want a secure and logical database with minimal errors. As users of the website, we’d want fast performance and a fair experience. Our project is mainly based on the popular competitive programming website **Codeforces** but with a thrilling added twist: *Battles*.

This report describes the **Dakowdas** main database design using an Entity-Relationship diagram. Section 3 mentions the tool(s) that were utilized to draw the ER. After that section 4 describes the requirements of the system. Following, section 5 is the legend of ER diagram symbols which aids in clarifying the ER. Furthermore, sections 6 and 7 encompass the entity types and relationships in detail. After that, section 8 goes step by step into the mapping algorithm, leading to section 9 which displays all tables. Section 9 illustrates how the tables work by providing table states, leading up to section 10 with the tables’ states. Thenceforth, the database’s implementation code is delineated in section 11. Finally, the report wraps up with a conclusion. By the end of this project, we aim to deliver a powerful and reliable database solution that will serve as the backbone of our competitive programming website, ensuring that

programmers can focus on what they do best: solving complex problems and pushing the boundaries of their coding abilities.

2-References/Copyright

Codeforces. (n.d.). https://codeforces.com

https://codedamn.com/news/sql/regular-expressions-in-sql

Legend Reference:

R. Elmasri and S.B. Navathe, "Fundamentals of Database Systems." Pearson, 2020, pp. 83. Figure 3.14: Summary of the notation for ER diagrams.

3- Tool(s) Used to Draw ER

Draw.io was the only tool used to draw the ER.

4- System Description & Requirements:

- A user chooses a unique username to be identified by, a password and a unique email used to login to the website. He fills in his first name, last name, and his country of residence, which can be selected from a predefined list. For each user, we must also record the registration date, the number of friends, and the number of problems solved successfully.
- Users can message each other at a given timestamp. Each message has some content.
- A user may also befriend another user at a certain time.
- A user can make an announcement. Each announcement is given a unique ID. It has some content which conveys some news or updates. It also has the language that it was written in (Eg: English, Russian,..), timestamp recording the time at which the announcement was made, and the time elapsed since the announcement.
- The competitive programming website also has a blog entry feature. A user may choose to post a blog entry which is displayed on his page. A distinctive ID is given to each and every blog entry, as well as a topic rating (vote from other users), a title, the timestamp at which it was posted, and

content of the entry. A blog entry can also include some hashtags that help other users explore blog entries in accordance with their preferences.

- A user comments on a blog entry to reflect his view on the topic discussed. The comment has content, a comment timestamp, and the vote it receives by viewers. Many users can comment on a blog entry.
- An admin is a user identified by an AdminID. Admins are responsible for maintaining standards and guidelines, contest coordination, user support, policy enforcement, etc. Such responsibilities of the admin must also be recorded, along with the contribution level of the admin. Furthermore, an admin has a role and can be a super admin, a moderator, or a support admin.
- The competitive programming website also has programming problems. Each problem is assigned a problem number and a difficulty level (A, B, C, D, E or F). The problem ID, composed of the problem number and difficulty level, sets a problem apart from others. A problem has a title, description, memory limit, time limit. Tags such as dynamic programming, bitmasks, and combinatorics are added to a problem to help categorize it and give insight into solving it.
- A solution entity is identified by the solution's ID and the specific problem which the solution solves. Each problem must have one and only one solution. The solution has source code.
- Each programming problem is tested by at least one test case. A test case is distinguished by the combination of the test case number and the problem ID. Each test case has input and expected output according to the problem specifications.
- Submissions are differentiated by the submission ID. For each submission, the programming language, source code, instance of submission (timestamp) and verdict (Accepted, Time Limit Exceeded on Test X, Memory Limit Exceeded on Test X, Wrong Answer on Test X) are recorded.
- A user may submit as many submissions to as many problems, but a specific submission is always submitted by one user only.
- The number of correct submissions must be recorded for every problem.
- The website also holds contests. A contest is characterized by a room number. A contest is also given a name, length, start timestamp, time before start, division (1,2,3, or 4), and short description to introduce people to the competition.
- A contest contains at least one programming problem.
- A programming problem used in one contest cannot be used by any other contest.
- A user may either be a contestant or a contest creator.
- No 2 contest creators share the same. Each contest creator organizes at least one contest, and every contest is organized by at least one contest creator.

- Each contestant is given an individualized ID. A contestant has a rating which reflects his competence in problem solving. Based on this rating, each contestant is assigned a bracket.
- A contestant must compete in at least one contest. For each contest the contestant takes part in, he gets a score, a rank (relative to other participating contestants), and the timestamp of registration. A contest is held even if no contestants compete in it.
- A contest creator may choose to notify a contestant regarding a programming problem. The notification content will be displayed and must be stored.
- A contestant can also hack a submission if he believes that it's incorrect and is invalid for a certain input. Each hack is assigned a distinguishing hack ID and has a timestamp, verdict (Invalid Input, Unsuccessful Hacking Attempt, or Successful Hacking Attempt), and test (for which the hacker believes the submission's code outputs a wrong answer).
- Another exciting feature is the battle. A battle represents a friendly one-to-one contest between 2 users. Only 2 users are allowed to join a battle, and the winner must be noted. A battle id is specified for every battle held, and each battle is composed of 2-3 programming problems.

5 - Legend of ER diagram symbols

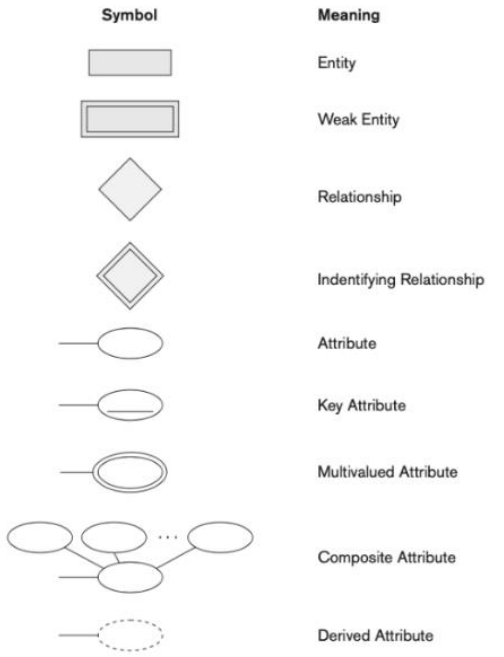
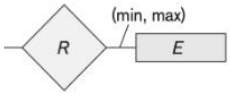
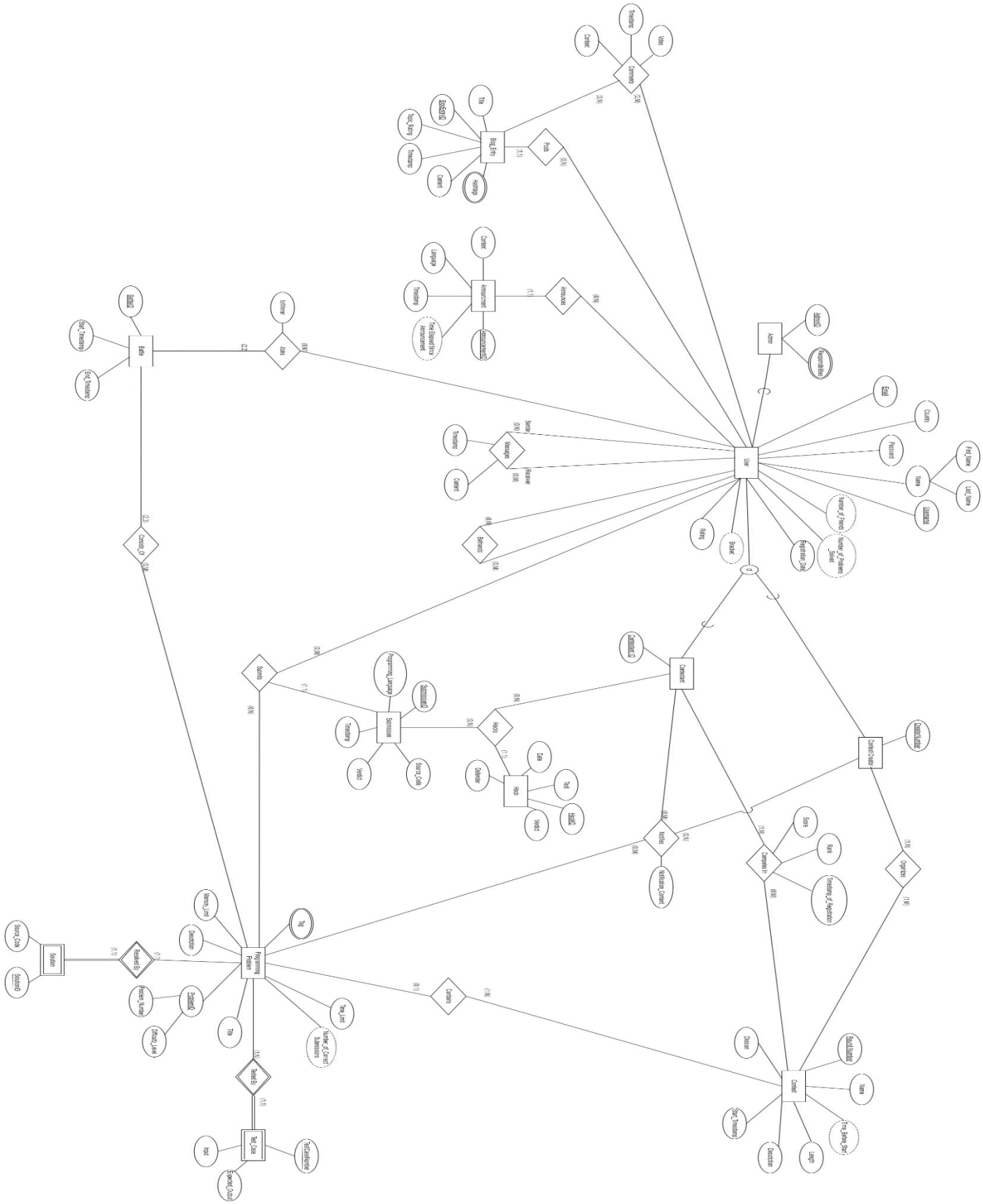


Figure 3.14
Summary of the notation for ER diagrams.

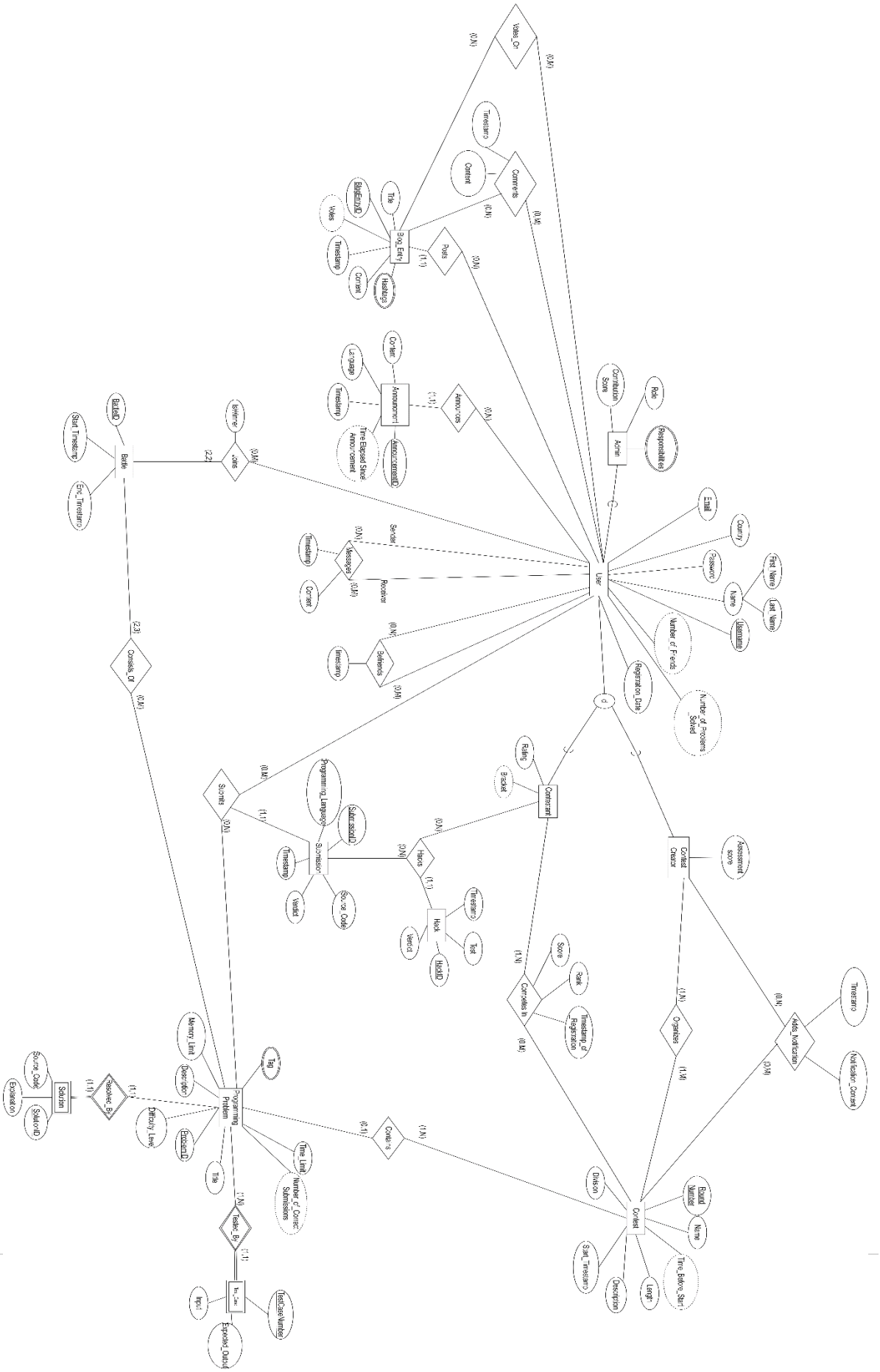


Structural Constraint (min, max) on Participation of *E* in *R*

6 – Complete Old ER Diagram for the Dakowdas database

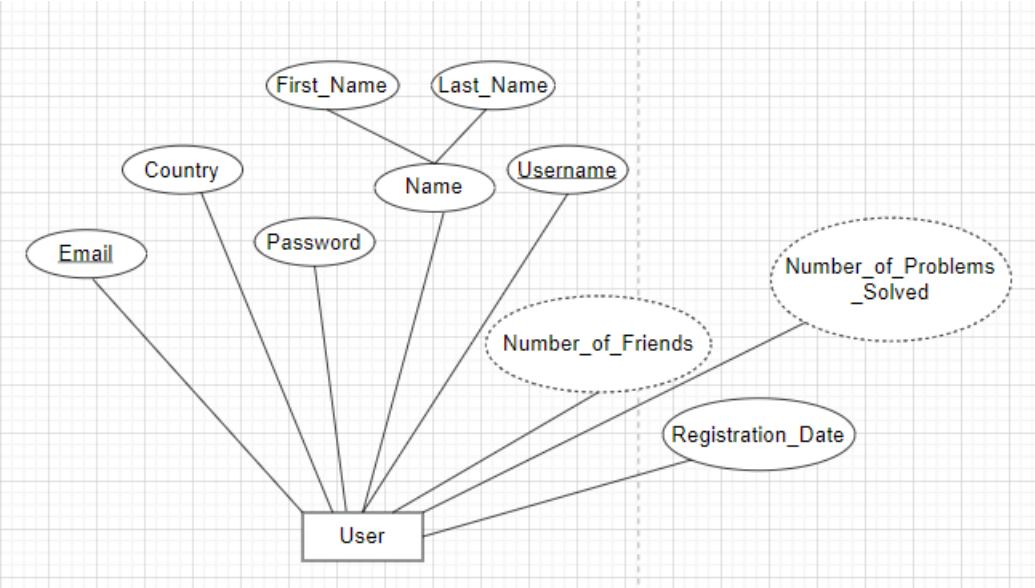


7 – Complete New ER Diagram for the Dakowdas database



7.1- Entity Types & Their Attributes

7.1.1- User:



-Entity Name: User

This entity represents all the people that use the competitive programming website. It includes the information displayed on the user's profile as well as the user's confidential information.

-Entity Type: strong

-Key Attribute(s): Username

>Reasoning Behind Choice of Key Attribute: A username may only be attributed to 1 user. Upon registration, the website prohibits the new user from using a username that's already taken

-Other Attributes:

1)Name: ***composite*** attribute. It is composed of the user's first name and the last name.

2)Email: used as contact information for the user. It can be used to notify the user of upcoming contests, rating updates, etc.

3)Password: used to validate that the party trying to register to the account is actually the alleged user.

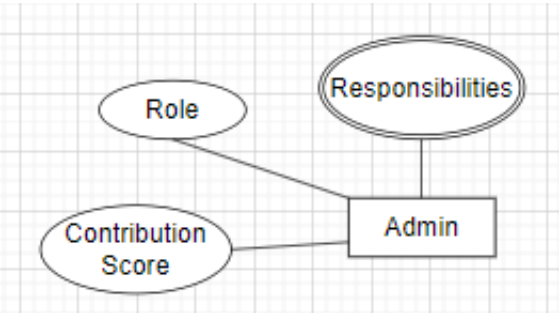
4)Country: country of residence of the user. It's used to rank users in the same region.

5)Registration Date

6)Number of Friends: ***derived*** attribute. It's derived from the Befriends relationship between 2 users.

7)Number of Problems Solved: ***derived*** attribute.It's derived from the number of correct submissions that the user submits to the programming problems.

7.1.2- Admin:



-Entity Name: Admin

Admins are those who run and manage the website, ensuring its overall functionality and adherence to a high standard.

-Entity Type: strong

-Key Attribute(s): AdminID

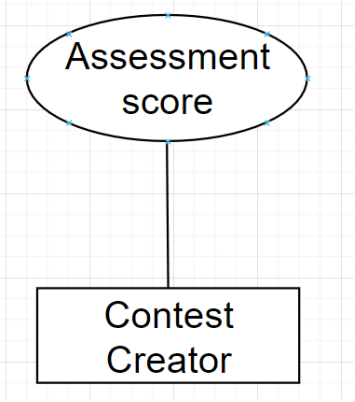
-Other Attributes:

1)Responsibilities: ***multivalued*** attribute. It's used to store the jobs assigned to an admin such as: user support, rating system, community guidelines, or technical maintenance. Since each admin may have several responsibilities, it's a ***multivalued*** attribute.

2) Admin Role: An attribute indicating the specific role or level of admin privileges (e.g., super admin, moderator, support, etc.). This can determine the scope of their authority.

3) Contribution Score: A custom score or metric to measure the admin's overall contributions to the Codeforces community.

7.1.3- Contest Creator:



-Entity Name: Contest Creator

Contests are organized by contest creators. This distinguishes a normal user from a contest creator.

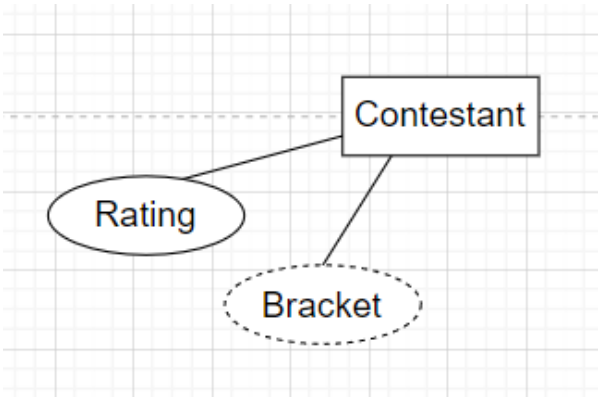
-Entity Type: strong

-Key Attribute(s): Creator Number

Each contest creator is given a unique creator number to distinguish him from other contest creators.

-Other Attributes: Assessment score evaluates the performance of the creator in terms of the quality, creativity, and originality of the contests he creates.

7.1.4- Contestant:



-Entity Name: Contestant

This entity represents those who compete in contests.

-Entity Type: strong

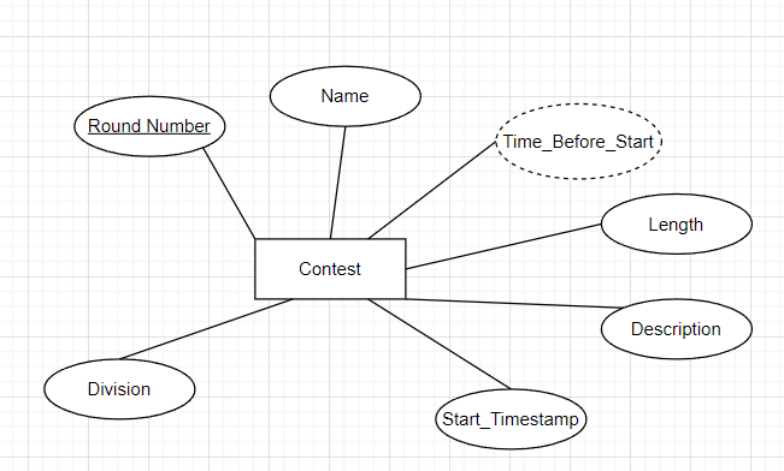
-Key Attribute(s): Contestant ID

Each contestant is assigned a unique contestant ID which helps in distinguishing contestants.

-Other attributes: 1) Rating: demonstrates the prowess of the contestant.

2) Bracket: ***derived*** attribute. Each bracket corresponds to a range of rating values. Brackets range from Newbie, Pupil, Specialist, Expert, Candidate Master, Grandmaster, etc. It's derived from the current rating of the contestant.

7.1.5- Contest:



-Entity Name: Contest

-Entity Type: strong

-Key Attribute(s): Round Number

-Other Attributes:

1)Name: considered as the title of the contest, often related to a theme, a sponsor, or a specific occasion.

2)Length: duration which the participants have to complete the contest, usually in hours and minutes.

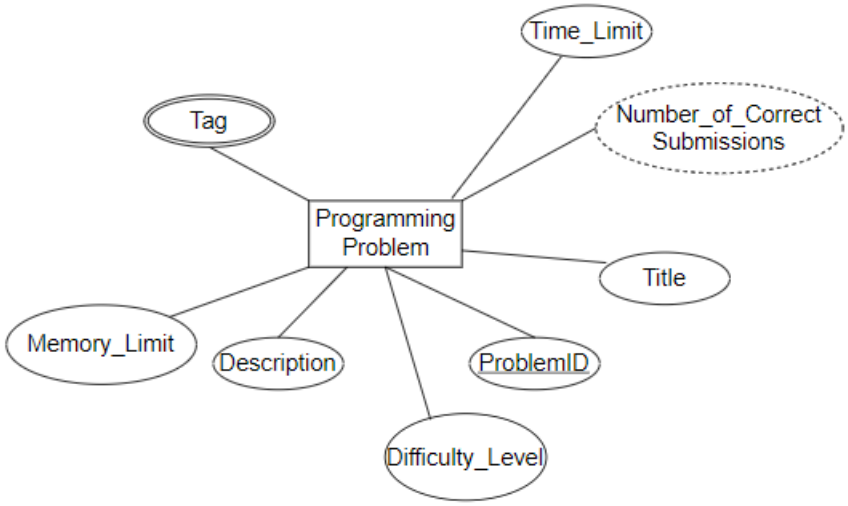
3)Start Timestamp: the date and time when the contest will begin. This is typically provided in Coordinated Universal Time (UTC).

4)Description: short introduction to the contest. This may include the inspiration or the min goals.

5)Division: this reflects the level of difficulty and expertise required of the competition. For example, division 1 is for more proficient and higher-rated users.

6)Time Before Start: ***derived*** attribute. It's derived from the current time and the start time of the competition. It helps contestants better prepare for the contest and prevents delayed entry.

7.1.6- Programming Problem:



-Entity Name: Problem

-Entity Type: strong

-Key Attribute(s): Problem ID

-Other Attributes:

1)Title

2)Description: introduction to the problem. It may contain a story to help elucidate the problem's logic.

3)Memory Limit: maximum memory that a submission may use. If a submission exceeds the memory limit, then it is rejected. It's usually given in gigabytes.

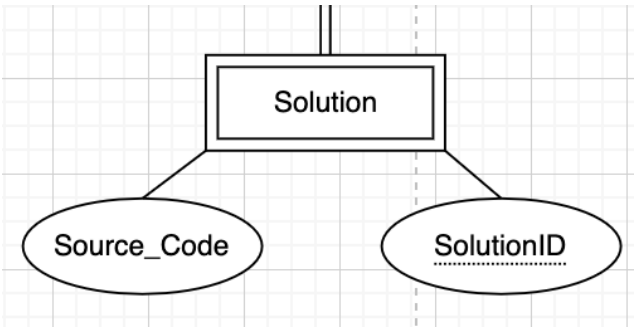
4)Time Limit: maximum running time that a submission may use. If a submission exceeds the time limit, then it is rejected. It's usually given in milliseconds.

5)Tag: ***multivalued*** attribute. Tags reflect the topics that the problem covers and that are required for a successful submission. These include: maths, greedy, dynamic programming, etc. It's ***multivalued*** since a single problem may cover several topics.

6)Number of Correct Submissions: derived attribute. It's derived from the number of submits relationship with submissions with an "Accepted" Verdict for the problem.

7) Difficulty Level: ex: A, B, C, D, E, F. A being the easiest, F being the toughest.

7.1.7- Solution:



-Entity Name: Solution

-Entity Type: weak. Its identifying entity is Programming Problem.

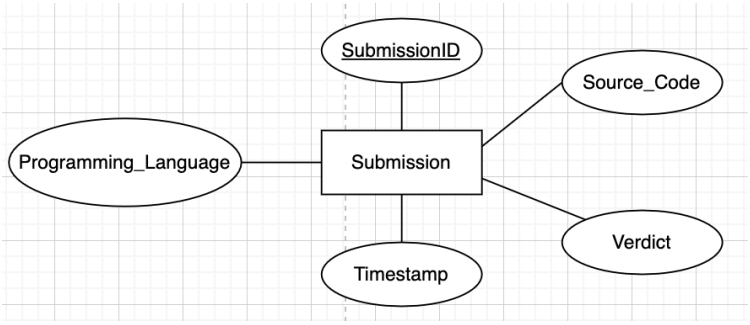
-Partial Key: Solution ID.

A solution is uniquely identified by the combination of the solution ID and the identifying programming problem's ID.

-Other Attributes:

1)Source Code: actual code implementation of the solution to the problem.

7.1.8- Submission:



-Entity Name: Submission

-Entity Type: strong

-Key Attribute(s): Submission ID

-Other Attributes:

1)Source Code: actual code implementation of the submission proposed by the user

2)Programming Language: the programming language used in the source code. This can be C++, Java, Python, or some other language.

3)Timestamp: exact instance of submission.

4)Verdict: this shows whether the submission was accepted or not.

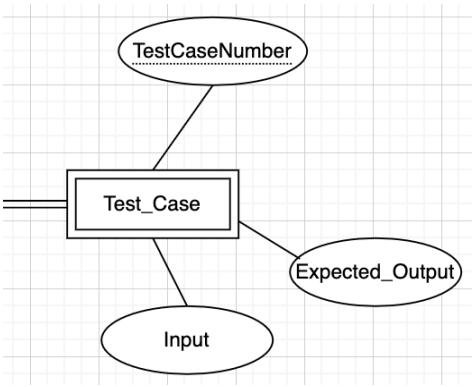
Verdicts are: -Accepted:source code ran correctly on all test cases.

-Wrong Answer on Test X: source code gave an output not equal to the expected output on test X.

-Time Limit Exceeded on Test X: source code exceeded the time limit when run on test X.

-Memory Limit Exceeded on Test X: source code exceeded the memory limit when run on test X.

7.1.9- Test Case:



-Entity Name: Test Case

-Entity Type: weak. Its identifying entity is Programming Problem.

-Partial Key: TestCaseNumber.

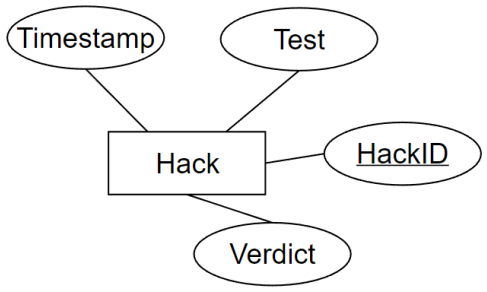
A solution is uniquely identified by the combination of the TestCaseNumeber and the identifying programming problem's ID.

-Other Attributes:

1)Input: input to the code

2)Expected output: correct output based on the input and problem specifications.

7.1.10- Hack:



-Entity Name: Hack

A contestant can hack another contestant's submission if he believes that the submission is faulty and gives a wrong output on a certain Test. If the hack is successful, the hacker is awarded some points. However, if the hack is unsuccessful, he will be penalized.

-Entity Type: strong

-Key Attribute(s): Hack ID

-Other Attributes:

1)Test: test that the hacker claims the submission fails on.

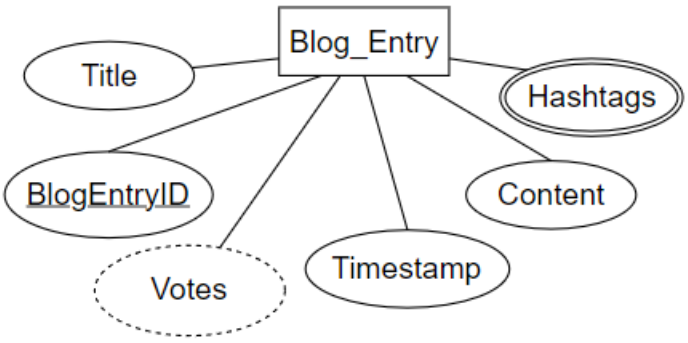
2)Verdict: -Invalid Input: input provided by the hacker doesn't meet the problem's input requirements.

-Unsuccessful Hacking Attempt: submission didn't fail on test.

-Successful Hacking Attempt submission failed on test.

3)Timestamp: date on which the hacker submitted the hack.

7.1.11- Blog Entry:



-Entity Name: Blog Entry

Blog Entry is used to give users a voice and allow them to share their concerns and ideas as well as interact with each other.

-Entity Type: strong

-Key Attribute(s): BlogEntryID

-Other Attributes:

1)Title: reflects the topic discussed in the entry.

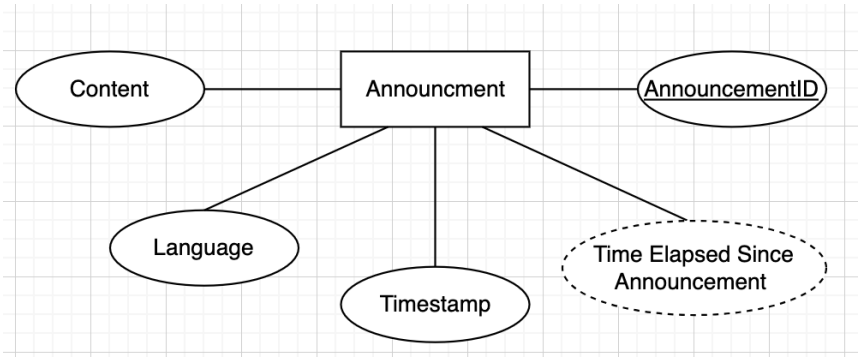
2)Content: actual text of the blog entry

3)Timestamp

3)Votes: derived attribute. Derived from the number of votes_on relationship.

4)Hashtags: ***multivalued*** attribute. Hashtags are used for categorization and discoverability of the blog entries posted by the users. They usually convey the theme of the blog. It's a ***multivalued*** attribute since a single blog entry may discuss several subject matters and hence have several hashtags.

7.1.12- Announcement:



-Entity Name: Announcement

Announcements are made by users to advertise contests.

-Entity Type: strong

-Key Attribute(s):AnnouncementID

-Other Attributes:

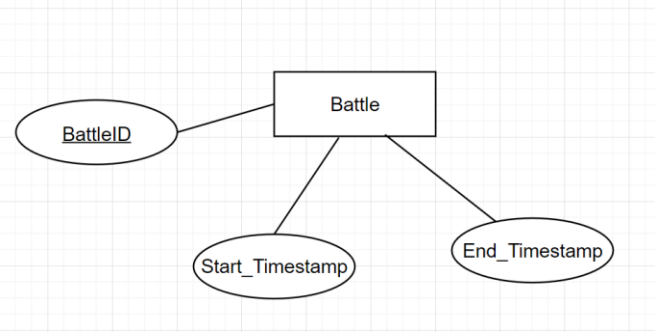
1)Content: actual text of the announcement

2)Language: language used in the announcement (for example: English, French, Russian)

3)Timestamp: time at which the announcement was made.

4)Time Elapsed Since Announcement: ***derived*** attribute. It shows how long it has been since the announcement was made. It's derived from the current time and the timestamp of the announcement.

7.1.13- Battle:



-Entity Name: Battle

Battles are similar to contests but on a more private level. They involve 2 users going head-to-head while solving 2-3 programming problems. The first to finish all problems successfully is crowned as the winner and is awarded extra points to his rating.

-Entity Type: strong

-Key Attribute(s):BattleID

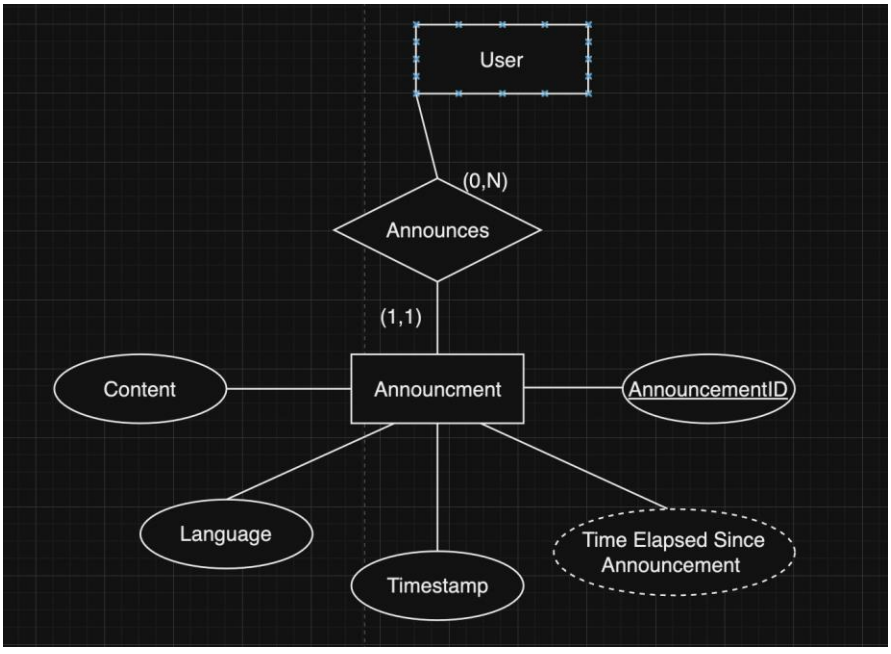
-Other Attributes:

1)Start Timestamp: time at which the battle started.

2)End Timestamp: time at which the battle ended. Note that a battle ends when the winner finishes all problems.

7.2- Relationships and Their Explanations

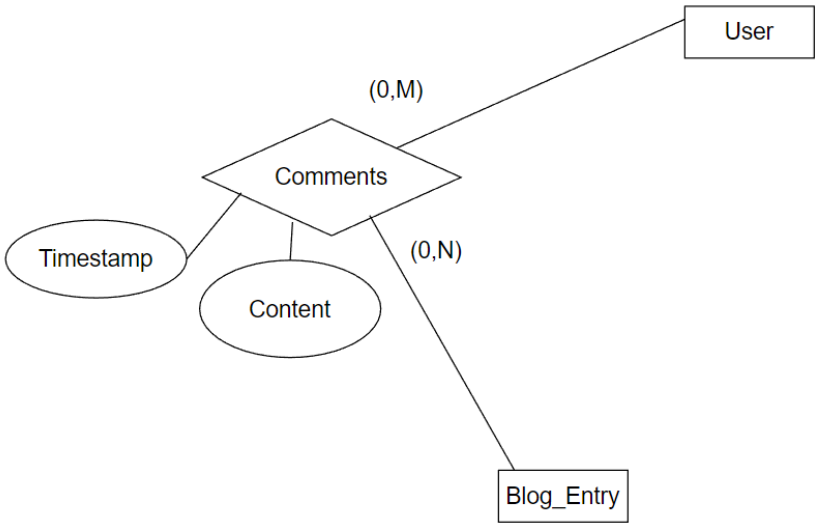
7.2.1- Announces



Relationship Type: Binary

A user can announce many announcements. However, an announcement can be announced by only one user.

7.2.2- Comments



Relationship Type: Binary

A user can comment in a Blog Entry.

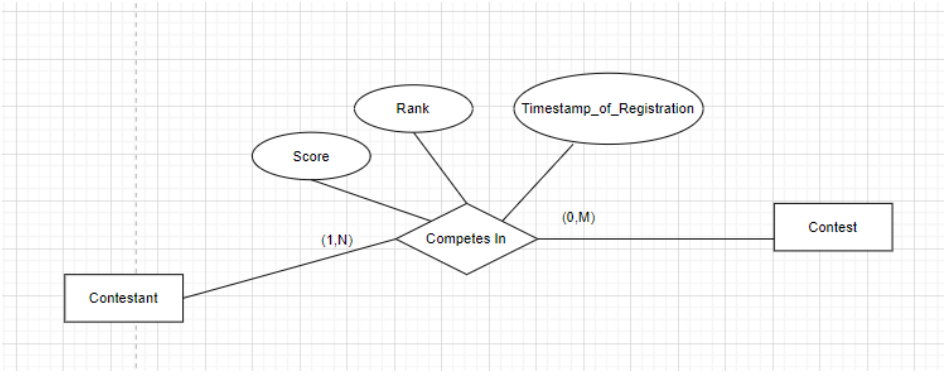
A Blog Entry can have many comments.

For each comment, we store:

-Content: information shared in the comment.

-Timestamp: time at which the comment was made

7.2.3 Competes



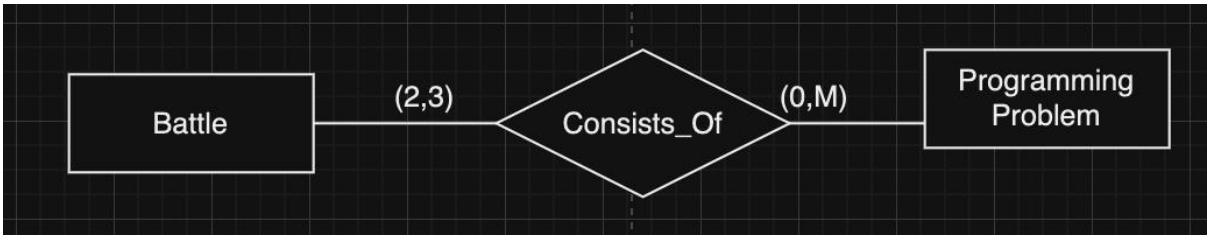
Relationship Type :Binary

A contestant must compete in at least one contest . A contest can have many contestants competing in it.

A contest will start even if no contestants compete in it.

The score, rank and timestamp of registration are recorded for each contestant in the competition.

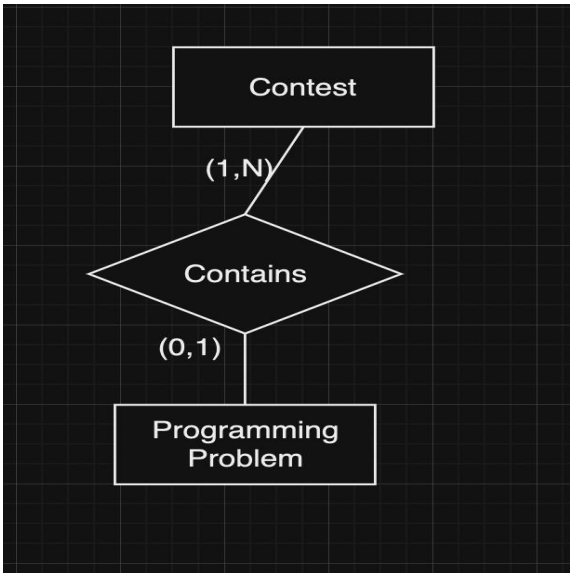
7.2.4 Consists Of



Relationship Type: Binary

A Battle consists of at least 2 and at most 3 problems. A programming problem can be part of many battles.

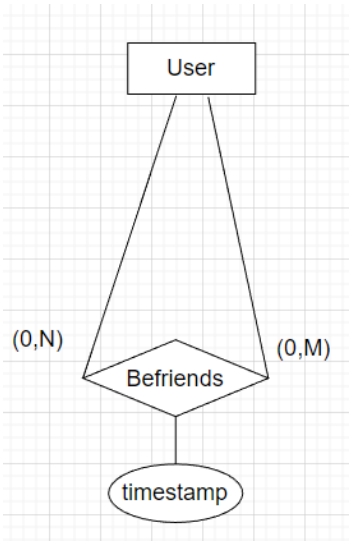
7.2.5- Consists Of



Relationship Type: Binary

A Contest contains at least one problem. A programming problem belongs to utmost one contest.

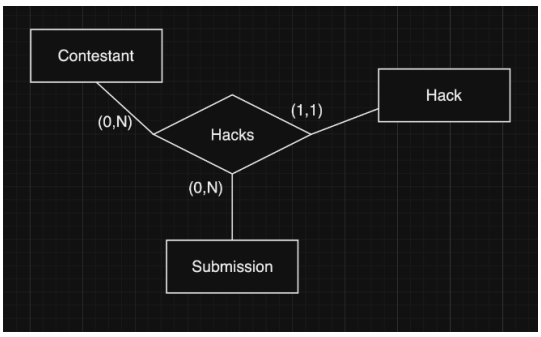
7.2.6- Befriends



Relationship Type: Recursive

A user can befriend many users. Timestamp of when the 2 users became friends is recorded.

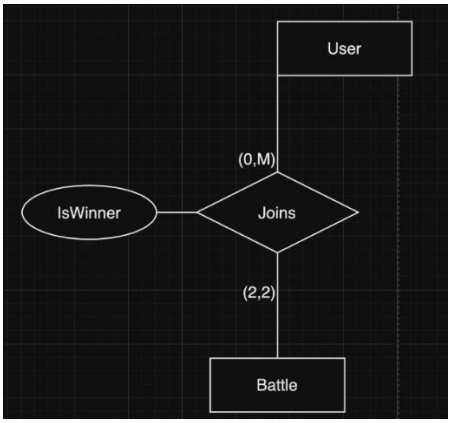
7.2.7- Hacks:



Relationship Type: Ternary

A contestant can hack many submissions with a hack. Any failed hacks are stored to penalise the hacker for an unsuccessful submission. Otherwise, the hacker is rewarded.

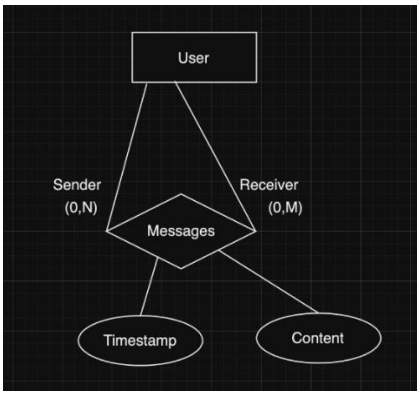
7.2.8- Joins:



Relationship Type: Binary

A user can join many battles. A battle consists of only 2 users.

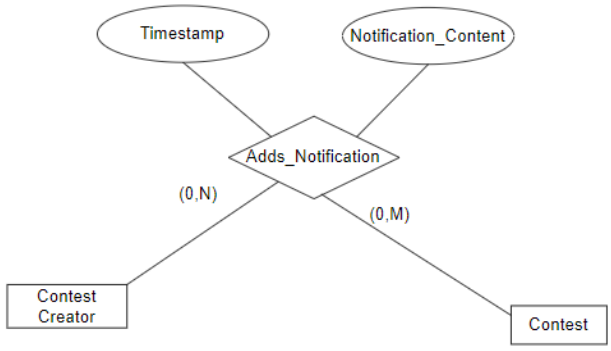
7.2.9- Messages:



Relationship Type: Recursive

A sender sends many messages. A receiver receives many messages. The content and timestamp of each message sent is recorded.

7.2.10- Adds_Notification:

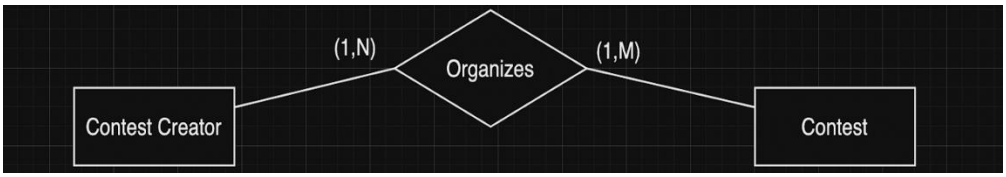


Relationship Type: Binary

A contestant creator can add a notification to a contest.

The content and timestamp of the notification are recorded.

7.2.11- Organizes:

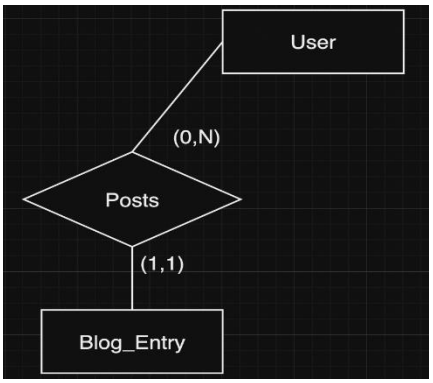


Relationship Type: Binary

A contest creator organizes at least one contest.

A contest must be organized by at least one contest creator.

7.2.12- Posts:

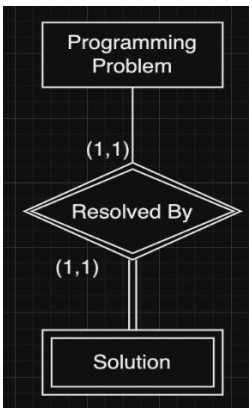


Relationship Type: Binary

A user can post many blog entries.

A blog entry must be posted by only one user.

7.2.13- Resolved By:

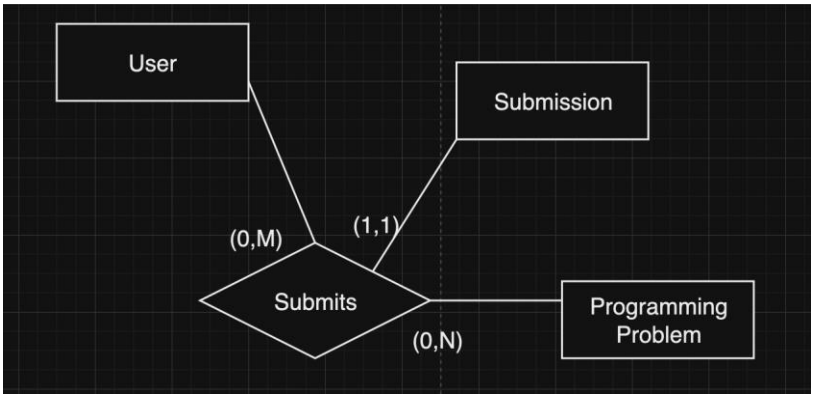


Relationship Type: Binary - Identifying Relationship

A programming problem is resolved by exactly one solution.

A solution resolves exactly one programming problem.

7.2.14- Submits:

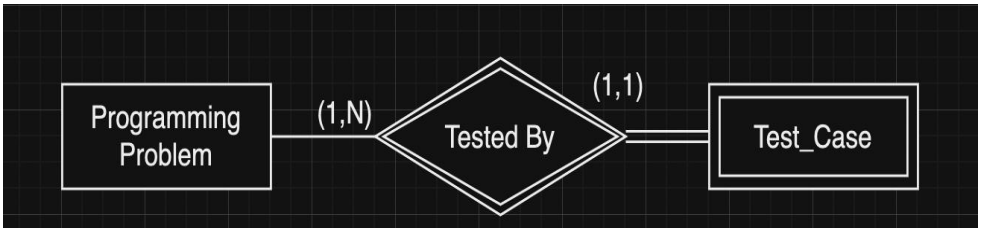


Relationship Type: Ternary

A user submits many submissions for many programming problems.

A submission is submitted by exactly one user.

7.2.15-Tested By:

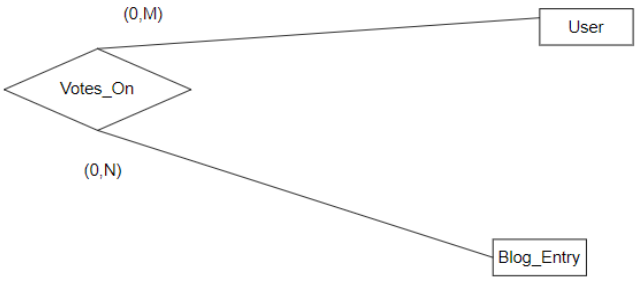


Relationship Type: Binary - Identifying Relationship

A Programming problem is tested by at least one test case.

A test case tests only one programming problem.

7.2.15-Votes_On:

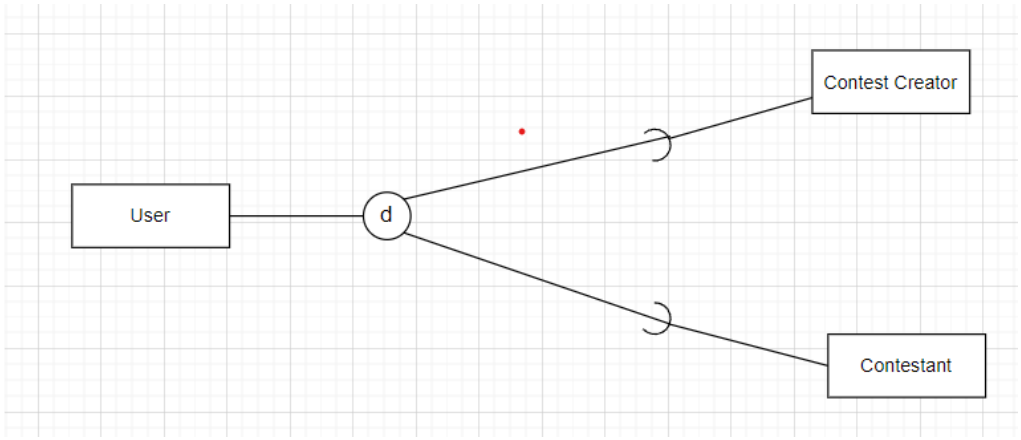


Relationship Type: Binary

A user may vote on several blog entries.

A blog entry may be voted on by multiple users.

7.2.16-Inheritance 1:

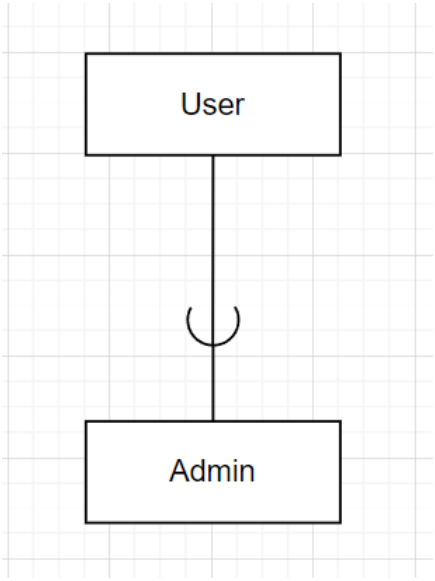


Parent: User

Children: Contestant & Contest Creator

Inheritance Type: Disjoint. A user can either be a contest creator or a contestant, but not both.

7.2.17-Inheritance 2:



Parent: User

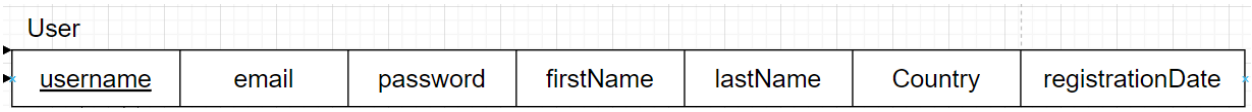
Children: Admin

An admin is a user with extra authority and responsibilities.

8 – ER to Relational Mapping:

8.1- Step 1 – Mapping Strong Entity Types

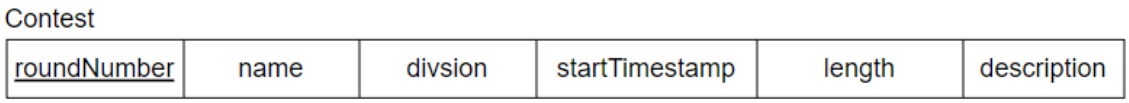
8.1.1-User:



Primary Key: username

Foreign Keys: None

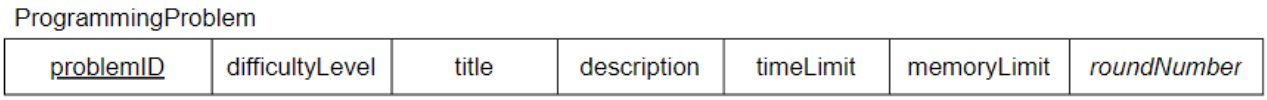
8.1.2-Contest:



Primary Key: roundNumber

Foreign Keys: None

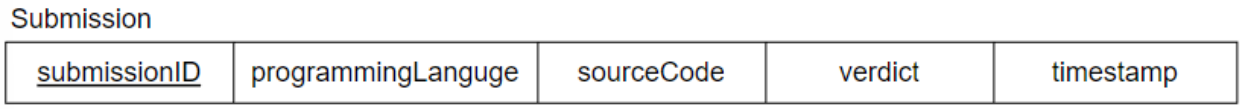
8.1.3-Programming Problem:



Primary Key: problemID

Foreign Keys: -roundNumber foreign key references roundNumber primary key in Contest relation.

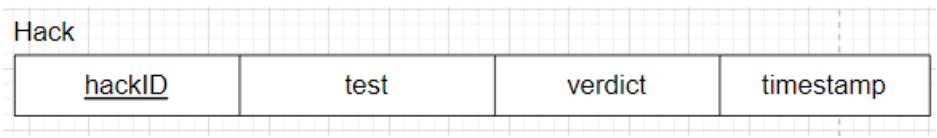
8.1.4-Submission:



Primary Key: submissionID

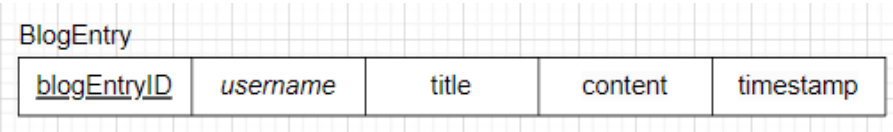
Foreign Keys: -None

8.1.4-Hack:



Primary Key: hackID

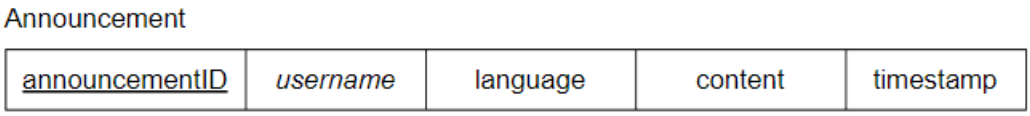
8.1.5-Blog Entry:



Primary Key: blogEntryID

Foreign Keys: -username foreign key references username primary key in User relation.

8.1.5-Announcement:



Primary Key: announcementID

Foreign Keys: -username foreign key references username primary key in User relation.

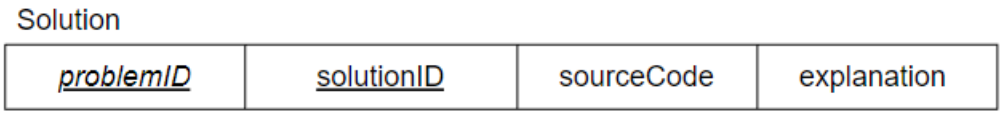
8.1.6-Battle:

Primary Key: battleID

Foreign Keys: -None

8.2 – Step 2 – Mapping of Weak Entity Types

8.2.1-Solution:



Primary Key: combination of solutionID (partial key of the weak entity type Solution) and problemID (primary key of the owner Programming Problem)

Foreign Keys: - problemID foreign key references problemID primary key in Programming Problem relation.

8.2.2-Test Case:

TestCase

problemID	testCaseNumber	input	expectedOutput

Primary Key: combination of testCaseNumber (partial key of the weak entity type TestCase) and problemID (primary of the owner entity Programming Problem)

Foreign Keys: - problemID foreign key references problemID primary key in Programming Problem relation.

8.3 – Step 3 – Mapping of Binary 1:1 Relationship Types

8.3.1-Resolved_By:

Solution

problemID	solutionID	sourceCode	explanation

1:1 Relationship between Programming Problem and Solution entity types.

Include primary key problemID of Programming Problem relation as a foreign key in Solution relation.

8.4 – Step 4 – Mapping of Binary 1:N Relationship Types

8.4.1-Tested_By:

TestCase

problemID	testCaseNumber	input	expectedOutput

1:N Relationship between Programming Problem and Test_Case entity types.

TestCase relation represents Test_Case entity type which participates at the N-side of the Tested_By relationship.

Include primary key problemID of ProgrammingProblem relation as a foreign key in TestCase relation.

8.4.2-Contains:

ProgrammingProblem

problemID	difficultyLevel	title	description	timeLimit	memoryLimit	*roundNumber*

1:N Relationship between Programming Problem and Contest entity types.
ProgrammingProblem relation represents Programming Problem entity type which participates at the N-side of the Tested_By relationship.
Include primary key roundNumber of Contest relation as a foreign key in ProgrammingProblem relation.

8.4.3-Posts:

BlogEntry

blogEntryID	*username*	conetent	timestamp	votes

1:N Relationship between User and Blog_Entry entity types.
BlogEntry relation represents Blog_Entry entity type which participates at the N-side of the Tested_By relationship.
Include primary key username of User relation as a foreign key in BlogEntry relation.

8.4.4-Announces:

Announcement

announcementID	*username*	language	content	timestamp

1:N Relationship between User and Announcement entity types.
Announcement relation represents Announcement entity type which participates at the N-side of the Tested_By relationship.
Include primary key username of User relation as a foreign key in Announcement relation.

8.5- Step 5 – Mapping of Binary M:N Relationship Types

8.5.1-Befriends:

Befriends

friend1Username	*friend2Username*	timestamp

M:N Recursive Relationship between 2 Users

Relation Befriends is created with the following:

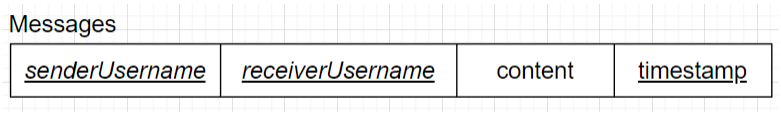
Primary Key: combination of friend1username and friend2username, the 2 usernames (primary keys) of the 2 User entities participating in the Befriends relationship.

Foreign Keys: -friend1Username references username primary key in User relation.

-frined2Username references username primary key in User relation.

Timestamp of befriends relationship is also recorded.

8.5.2-Messages:



M:N Recursive Relationship between 2 Users

Relation Messages is created with the following:

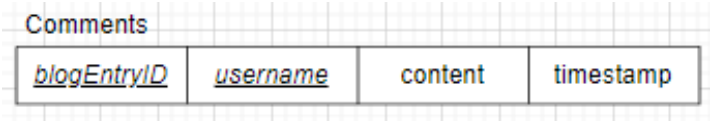
Primary Key: combination of timestamp along with senderUsername and receiverUsername, the 2 usernames (primary keys) of the 2 User entities participating in the Messages relationship, .

Foreign Keys: - senderUsername references username primary key in User relation.

- receiverUsername references username primary key in User relation.

Content of message relationship is also recorded.

8.5.3-Comments:



M:N Relationship between User and BlogEntry

Relation Comments is created with the following:

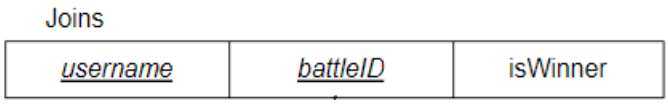
Primary Key: combination of blogEntryID and username, the primary keys of the BlogEntry and User entities participating in Comments relationship.

Foreign Keys: - blogEntryID references blogEntryID primary key in BlogEntry relation.

- username references username primary key in User relation.

Content, timestamp and votes of the comment relationship are also recorded.

8.5.4-Joins:



M:N Relationship between User and Battle

Relation Joins is created with the following:

Primary Key: combination of battleID and username, the primary keys of the Battle and User entities participating in Joins relationship.

Foreign Keys: - battleID references battleID primary key in Battle relation.

- username references username primary key in User relation.

isWinner boolean statues of the joins relationship is also recorded.

8.5.5-ConsistsOf:

ConsistsOf

battleID	problemID

M:N Relationship between Programming Problem and Battle

Relation ConsistsOf is created with the following:

Primary Key: combination of battleID and problemID, the primary keys of the Battle and Programming Problem entities participating in ConsistsOf relationship.

Foreign Keys: - battleID references battleID primary key in Battle relation.

- problemID references problemID primary key in Programming Problem relation.

8.5.6-Organizes:

Organizes

creatorUsername	roundNumber

M:N Relationship between ContestCreator and Contest

Relation Organizes is created with the following:

Primary Key: combination of creatorUsername and roundNumber, the primary keys of the ContestCreator and Contest entities participating in Organizes relationship.

Foreign Keys: - creatorUsername references creatorUsername primary key in ContestCreator relation.

- roundNumber references roundNumber primary key in Contest relation.

8.5.7- CompetesIn:

CompetesIn

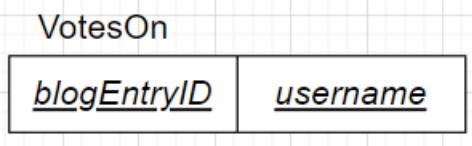
contestantUsername	roundNumber	score	rank	timestampOfRegistration

M:N Relationship between Contestant and Contest

Relation CompetesIn is created with the following:

Primary Key: combination of contestantUsername and roundNumber, the primary keys of the Contestant and Contest entities participating in CompetesIn relationship.
Foreign Keys: - contestantUsername references contestantUsername primary key in Contestant relation.
- roundNumber references roundNumber primary key in Contest relation.

8.5.8- VotesOn:

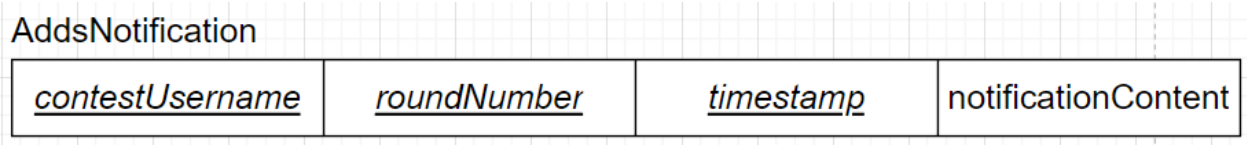


M:N Relationship between User and BlogEntry

Relation VotesOn is created with the following:

Primary Key: combination of blogEntryID and username, the primary keys of the BlogEntry and User entities participating in VotesOn relationship.
Foreign Keys: - username references username primary key in User relation.
- blogEntryID references blogEntryID primary key in BlogEntry relation.

8.5.9- AddsNotification:



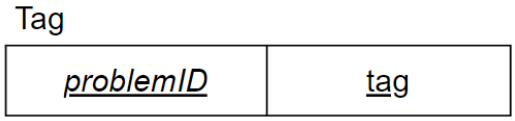
M:N Relationship between ContestCreator and Contest

Relation AddsNotification is created with the following:

Primary Key: combination of contestUsername and roundNumber, the primary keys of the ContestCreator and User entities participating in Contest relationship.
Foreign Keys: - contestUsername references contestUsername primary key in ContestCreator relation.
- roundNumber references roundNumber primary key in Contest relation

8.6- Step 6 – Mapping of multivalued attributes

8.6.1- Tag:



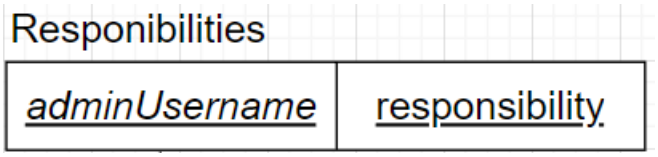
Multivalued attribute of Programming Problem entity.

Relation Tag is created with the following:

Primary Key: combination of problemID which is the primary key of the Programming Problem relation and tag.

Foreign Keys: - problemID references problemID primary key in Programming Problem relation.

8.6.2- Responsibilities:



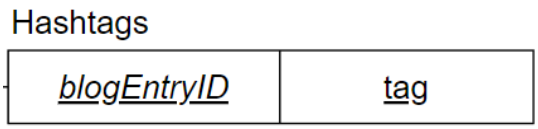
Multivalued attribute of Admin entity.

Relation Responsibilities is created with the following:

Primary Key: combination of adminUsername which is the primary key of the Admin relation and responsibility.

Foreign Keys: - adminUsername references adminUsername primary key in Admin relation.

8.6.3- Hashtags:



Multivalued attribute of BlogEntry entity.

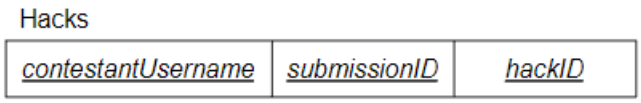
Relation Hashtags is created with the following:

Primary Key: combination of blogEntryID which is the primary key of the BlogEntry relation and tag.

Foreign Keys: - blogEntryID references blogEntryID primary key in BlogEntry relation.

8.7- Step 7 – Mapping of N-ary attributes

8.7.1- Hacks:



Ternary relationship between Contestant, Submission, & Hack.

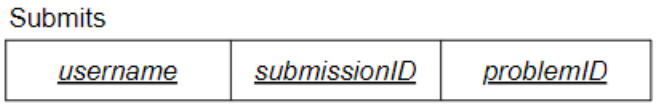
Relation Hacks is created as follows:

Primary Key: combination of contestantUsername which is the primary key of the Contestant relation, submissionId which is the primary key of the Submission relation, and hackID which is the primary key of the Hack relation.

Foreign Keys: - contestantUsername references contestantUsername primary key in Contestant relation.

- submissionId references submissionId primary key in Submission relation.
- hackID references hackID primary key in Hack relation.

8.7.2- Submits:



Ternary relationship between User, Submission, & Programming Problem.

Relation Submits is created as follows:

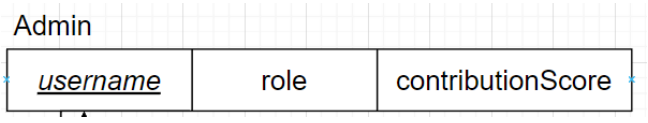
Primary Key: combination of username which is the primary key of the User relation, submissionId which is the primary key of the Submission relation, and problemID which is the primary key of the ProgrammingProblem relation.

Foreign Keys: - username references username primary key in User relation.

- submissionId references submissionId primary key in Submission relation.
- problemID references problemID primary key in ProgrammingProblem relation.

8.8- Step 8 – Mapping aggregation, specialization/generalization relationships

8.8.1- Admin:



Admin inherits from User.

Primary Key: -username (inherited username from User)

Foreign Keys: - username references username primary key in User relation.

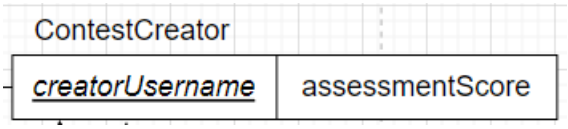
The role and contribution score are also recorded.

8.8.1- ContestCreator and Contestant:

ContestCreator and Contestant are both subclasses of the superclass User. They must be disjoint.

(Disjoint, partial specialization)

8.8.1.1. ContestCreator

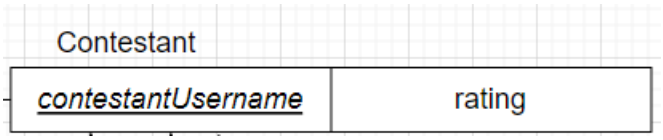


Primary Key: -creatorUsername (inherited username from User)

Foreign Keys: - creatorUsername references username primary key in User relation.

The assessment score is also recorded.

8.8.1.1. Contestant

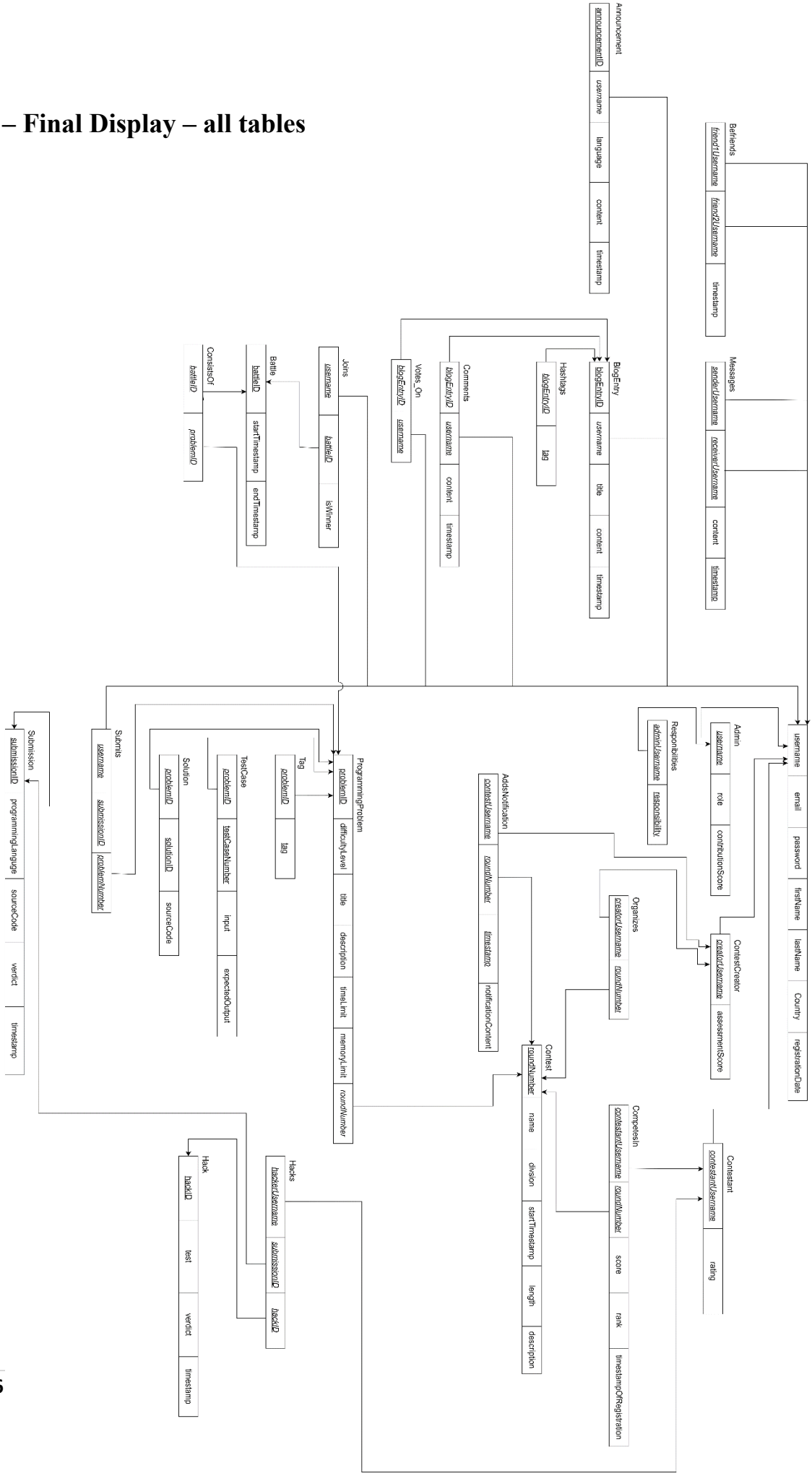


Primary Key: -contestantUsername (inherited username from User)

Foreign Keys: - contestantUsername references username primary key in User relation.

The rating is also recorded.

9 – Final Display – all tables



10-Tables' States

User

Username	email	password	firstName	lastName	Country	registrationDate
Khaled47	Khaled47@gmail.com	DamanWizDaPlan	Khaled	Charaf	Lebanon	1/1/2024
ay_sharaf	ay_sharaf@gmail.com	ay_s098	Ayman	Charaf	Lebanon	1/2/2024
AliFarhat	AliFarhat@gmail.com	Alifa987	Ali	Farhat	Lebanon	1/3/2024
Dankosay	Dankosay@gmail.com	Dana789	Dana	Kossaybati	Lebanon	1/4/2024
Brokie	Brokie@gmail.com	Brokie654	Joseph	Aoun	Lebanon	1/5/2024
Tourist	Tourist@gmail.com	Tourist456	Gennady	Korotkevich	Belarus	1/6/2024
jiangly	jiangly@gmail.com	jiangly321	Lingyu	Jiang	China	1/7/2024
Benq	Benq@gmail.com	Benq123	Benjamin	Qi	United States	1/8/2024
HussBak	HussBak@gmail.com	DakowdezTheBest	Hussein	Bakri	Lebanon	1/9/2024
ImanGha	ImanGha@gmail.com	KhaledIsMyFavorite	Iman	Ghalayini	Lebanon	1/10/2024

Contest

roundNumber	Division	name	description	startTimestamp	length
1	3	PinelyRounds	In pinely , we believe. Yes you know it, winner winner chicken dinner.	2023-10-18 09:23:45	2:00:00
2	3	HarbourSpace	Don't you just love HarbourSpace university. Earn a free scholarship by placing top 1 on our comptetion	2023-08-15 14:37:22	2:30:00
3	1	AUB CUP	DA BEST university in the region, You know it. Earn a free scholarship by placing top 10 on our coding competition	2023-05-27 19:51:10	2:00:00
4	1	Face CUP	Get a free job interver at facebook by placing top 50 on our hacker cup	2023-04-02 08:12:30	2:00:00
5	1	MIT CUP	The legendary umass competition is up. Place top one to get accepted at the best university in the world	2023-11-09 17:29:55	2:30:00
6	3	Educational Round	Educational Round on Dakowdez website	2023-06-30 11:45:18	2:30:00
7	3	Educational Round	Educational Round on Dakowdez website	2023-02-14 22:03:40	3:30:00
8	2	Hussein Bakri CUP	The best database teacher here. Place top 5 on my coding contest and you will get 2500$ in my database company !!! Capiche ?	2023-07-20 05:55:08	3:30:00
9	1	Iman Ghalayini CUP	Place top 10 in my hacker cup and you will receive a free skip lab offer on Bakri's next 433 database class	2023-09-03 13:18:57	4:00:00
10	1	KhaledTheBest CUP	I mean there are no prizes in my competition because we all know Khaled is top 1 :) !	2023-12-25 00:00:00	3:30:00

Befriends

friend1Username	friend2Username	TimeStamp
Khaled47	AliFarhat	2023-03-12 16:40:25
HussBak	ImanGha	2023-06-19 03:27:15
Khaled47	HussBak	2023-10-05 12:08:38
Khaled47	ImanGha	2023-04-30 08:56:50
ay_sharaf	Dankosay	2023-07-14 20:17:09
Brokie	Tourist	2023-09-28 06:45:33
HussBak	Khaled47	2023-01-21 14:59:42
ImanGha	Khaled47	2023-08-08 10:30:07
HussBak	ay_sharaf	2023-11-11 19:22:55
HussBak	Dankosay	2023-05-07 02:33:11

Message

senderUsername	recieverUsername	Content	TimeStamp
Khaled47	AliFarhat	Yo Bestie Wassup	2023-08-21 17:12:04
HussBak	ImanGha	This student Khaled of mine is t	2023-02-03 09:47:32
Khaled47	HussBak	Hello Proffesor Bakri	2023-06-13 22:30:55
Khaled47	ImanGha	Hello Miss Iman	2023-10-29 14:25:19
ay_sharaf	Dankosay	yo wassup	2023-04-15 06:18:40
Brokie	Tourist	OMG Tourist wassup	2023-12-07 11:55:03
HussBak	Khaled47	Good job on that 1000 rating ma	2023-03-26 19:37:28
ImanGha	Khaled47	Hello Khaled, Congrats on your r	2023-09-17 03:03:11
HussBak	ay_sharaf	Congrats on getting that 1 point	2023-07-05 15:49:29
HussBak	Dankosay	Comeon Dana, you should partic	2023-01-01 00:00:00

Submission

SubmissionID	programmingLangu	sourceCode	verdict	timeStamp	Username	problemNumber
1	C++	#include <iostream>	accepted	2023-04-20 08:14:52	Khaled47	1
2	C++	#include <iostream>	Wrong Answer	2023-07-11 17:26:39	ay_sharaf	2
3	C++	#include <iostream>	Compilation Error	2023-11-28 03:59:18	AliFarhat	3
4	Python	#!/usr/bin/env python3	Time Limit Exceeed	2023-05-14 12:33:27	Dankosay	4
5	C++	#include <iostream>	accepted	2023-09-05 21:45:01	Brokie	5
6	C++	#include <iostream>	accepted	2023-01-19 06:02:45	Tourist	6
7	C++	#include <iostream>	Memory Limit Exceeded	2023-06-07 14:57:10	jiangly	7
8	C++	#include <iostream>	Wrong Answer	2023-10-02 00:25:33	Benq	8
9	Java	public class AddOneToTwo {	Wrong Answer	2023-03-23 10:09:47	HussBak	9
10	Python	import random	accepted	2023-08-16 19:41:54	ImanGha	10
11	C++	#include <iostream>	accepted	2023-03-23 10:09:47	Khaled47	11

Programming Problem

problemID	Tag	difficultyLevel	title	description	timeLimit	memoryLimit	roundNumber
1	A	800	Sum of Three	Monocarp has an integer n. He wants to represent his number as a sum of three distinct positive integers x,y, and z. Additionally, Monocarp wants none of the	1	256	1
2	B	1400	Three Threadlets	Once upon a time, Decim found three threadlets and a pair of scissors. In one operation, Decim chooses any threadlet and cuts it into two threadlets, whose	1	512	1
3	C	2000	Decreasing String	Recall that string a is lexicographically smaller than string b if a is a prefix of b (and a ≠ b), or there exists an index i (1 ≤ i ≤ min(|a|, |b|)) such that ai < bi, and	2	128	1
4	A	1000	Rigged!	There are athletes participating in the competition, with the -th athlete having strength and endurance . The 1st athlete is Monocarp's friend Polycarp, and	3	256	2
5	B	1500	Aleksa and Stack	After the Serbian Informatics Olympiad, Aleksa was very sad because he didn't win a medal (he didn't know stack). So, Vasilije came to give him an easy	2	512	2
6	C	2200	Jellyfish and EVA	Monsters have invaded the town again! Asuka invites her good friend, Jellyfish, to drive EVA with her.	1	128	2
7	A	500	Short Sort	There are three cards with letters , , placed in a row in some order. You can do the following operation at most once:	2	256	3
8	B	1000	Good Kid	Slavic is preparing a present for a friend's birthday. He has an array a of n digits and the present will be the product of all these digits. Because Slavic	3	512	3
9	C	1500	Non-coprime Split	You are given two integers l ≤ r. You need to find positive integers a and b such that the following conditions are simultaneously satisfied:	4	128	3
10	A	1200	MEXanized Array	You are given three non-negative integers n, k & x. Find the maximum possible sum of elements in an array consisting of non-negative integers, which has	3	256	4
11	B	1600	Friendly Arrays	You are given two arrays of integers — a_1, ..., ab of length n, and b_1, ..., bm of length . You can choose any element ¡ from array (1 ≤ ≤), and for all 1 ≤	1	512	4
12	C	2200	Salyg1n and the MEX Game	This is an interactive problem!salyg1n gave Alice a set S of distinct integers a1, s2,...,sn (0 ≤ si ≤ 10^9). Alice decided to play a game with this set against Bob. The	1	128	4
13	A	1600	Ambitious Kid	Chaneka, Pak Chanek's child, is an ambitious kid, so Pak Chanek gives her the following problem to test her ambition.	2	256	5
14	B	2000	XOR Palindromes	You are given a binary string of length (a string that consists only of 0 and 1). A number is good if there exists a binary string of length , containing ones,	3	512	5
15	C	2400	Word on the Paper	On an 8×8 grid of dots, a word consisting of lowercase Latin letters is written vertically in one column, from top to bottom.	2	128	5
16	A	750	To My Critics	Suneet has three digits a, b, and c. Since math isn't his strongest point, he asks you to determine if you can choose	1	256	6
17	B	1250	Ten Words of Wisdom	In the game show "Ten Words of Wisdom," there are participants numbered from 1 to , each of whom submits one response. The -th response is words	2	512	6
18	C	1700	Tiles Comeback	Vlad remembered that he had a series of n tiles and a number k. The tiles were numbered from left to right, and the i-th tile had color ci.	3	128	6
19	A	750	Morning Sandwich	Monocarp always starts his morning with a classic sandwich. Sandwiches Monocarp makes always consist of bread, cheese, and/or ham.	4	256	7
20	B	1200	Come Together	Bob and Carol spent the whole day with Alice, but now it's time to go home. Alice, Bob, and Carol live on an infinite 2D grid in cells , , and , respectively.	3	512	7
21	C	1600	Tenzing and Balls	Tenzing has n balls arranged in a line. The color of the i-th ball from the left is ai.Tenzing can do the following operation any number of times:	1	128	7
22	A	950	Unit Array	Given an array a of length n, which elements are equal to −1 and 1. Let's call the array good if the following conditions are held at the same time:	1	256	8
23	B	1350	Maximum Strength	Fedya is playing a new game called "The Legend of Link", in which one of the character's abilities is to combine two materials into one weapon. Each material	2	512	8
24	C	1700	Travel Plan	During the summer vacation after Zhongkao examination, Tom and Daniel are planning to go traveling.	3	128	8
25	A	1850	Musical Puzzle	Vlad decided to compose a melody on his guitar. Let's represent the melody as a sequence of notes corresponding to the characters 'a', 'b', 'c', 'd', 'e', 'f', and 'g'.	2	256	9
26	B	2000	Rudolph and Tic-Tac-Toe	Rudolph invented the game of tic-tac-toe for three players. It has classic rules, except for the third player who plays with pluses. Rudolf has a 3×3 field — the	1	512	9
27	C	2600	Rudolf and the Another Comp	Rudolf has registered for a programming competition that will follow the rules of ICPC. The rules imply that for each solved problem, a participant gets 1 point,	2	128	9
28	A	2100	Ian Visits Mary	Ian and Mary are frogs living on lattice points of the Cartesian coordinate plane, with Ian living on (0,0) and Mary living on (a,b).	3	256	10
29	B	2500	Grid Reconstruction	Consider a 2×n grid, where n is an even integer. You may place the integers 1, 2, ..., 2 on the grid, using each integer exactly once.	4	512	10
30	C	3000	Similar Polynomials	A polynomial A(x) of degree d is an expression of the form A(x)= $a_0 + a_1x + a_2x^2 +$ + adx⊃, whre are integers, and ⊃ ≠ 0. Two polynomials ( ) and ( ) are	3	128	10

Battle

BattleID	startTimeStamp	endTimeStamp
1	2023-12-12 19:48:26	023-02-28 16:45:20
2	2023-06-05 11:09:33	2023-09-11 10:18:35
3	2023-03-28 08:37:14	2023-06-24 03:54:42
4	2023-09-19 16:20:57	2023-12-05 21:09:57
5	2023-01-15 04:52:42	2023-04-16 14:27:10
6	2023-07-22 14:14:30	2023-10-20 06:33:25
7	2023-10-31 00:00:00	2023-07-09 19:46:30
8	2023-04-08 05:26:03	2023-03-13 12:01:45

Joins

Username	battileID	isWinner
Khaled47	1	TRUE
Dankosay	1	FALSE
Khaled47	2	TRUE
ay_sharaf	2	FALSE
Khaled47	3	FALSE
HussBak	3	TRUE
HussBak	4	TRUE
ImanGha	4	FALSE
ay_sharaf	5	FALSE
Dankosay	5	TRUE

Consists Of

battleID	problemID
1	1
1	2
2	3
2	4
3	5
3	6
4	7
4	8
5	9
5	10

Test Cases

problemID	testCaseNumber	input	expectedO
1	1	10	13
2	1	11	14
3	1	12	15
4	1	13	16
5	1	14	17
6	1	15	18
7	1	16	19
8	1	17	20
9	1	18	21
10	1	19	22
11	1	20	23
12	1	21	24
13	1	22	25
14	1	23	26
15	1	24	27
16	1	25	28
17	1	26	29
18	1	27	30
19	1	28	31
20	1	29	32
21	1	30	33
22	1	31	34
23	1	32	35
24	1	33	36
25	1	34	37
26	1	35	38
27	1	36	39
28	1	37	40
29	1	38	41
30	1	39	42

Tag

problemID	tag
1	Math
1	DP
2	Math
2	DP
2	Data Structures
6	Math
7	DP
8	Data Structures
9	Bitwise XOR
9	Bitwise XOR
10	DakoDPwdez

Hashtags

blogEntryID	Hashtags
1	pro
2	coding
3	AliFarhat
4	HIIIII
5	Dynamic
6	Competitive
7	Dakowdez
8	Dakowdez
9	Khaled47
9	The Best
10	Dakowdez

Contestant

Username	Rating
Tourist	2123
jiangly	2000
Benq	1750
ay_sharaf	1500
AliFarhat	1250
Khaled47	1000
Dankosay	0

Announcements

announcemen	username	language	content	timestamp
1	Khaled47	English	Hello Dacowders...	2023-03-02 08:15:30
2	ay_sharaf	English	Hello Dacowders...	2023-07-19 14:46:42
3	AliFarhat	English	Hello Dacowders...	2023-11-10 21:28:55
4	Dankosay	English	Hello Dacowders...	2023-05-28 03:57:10
5	Brokie	English	Hello Dacowders...	2023-09-15 10:33:22
6	Tourist	English	Hello Dacowders...	2023-02-07 17:12:41
7	jiangly	English	Hello Dacowders...	2023-06-14 00:00:00
8	Benq	English	Hello Dacowders...	2023-10-01 06:45:18
9	HussBak	English	Hello Dacowders...	2023-04-25 13:22:29
10	ImanGha	English	Hello Dacowders...	2023-08-31 19:59:54

BlogEntry

blogEntryID	username	content	timestamp	votes
1	Khaled47	How to prevent hack	2023-10-18 08:45:23	5
2	ay_sharaf	Plan to get 2000 ratir	2023-09-25 16:30:10	4
3	AliFarhat	I am a pro player	2023-08-12 12:15:47	6
4	Dankosay	HIIIIIIII Wassuppp W	2023-07-03 19:55:33	2
5	Brokie	Eyo wassup today w	2022-06-14 05:20:55	1
6	Tourist	Welcome to my blog	2023-05-29 14:10:38	5
7	jiangly	Welcome to my blog	2023-04-22 03:05:12	4
8	Benq	Welcome to my blog	2023-03-17 09:28:40	3
9	HussBak	This is the best codi	2023-02-09 18:40:25	3
10	ImanGha	This is the best codi	2023-01-04 20:11:17	2

Comments

blogEntryID	username	content	timeStamp	votes
1	Khaled47	So just don't hack	2023-12-15 14:22:30	3
2	ay_sharaf	Practice !!!	2023-11-20 10:55:45	6
3	AliFarhat	Yeah so don't try	2023-10-07 17:30:18	5
4	Dankosay	HIIII	2023-09-03 22:40:55	2
5	Brokie	so we gonna talk ab	2023-08-28 08:15:27	3
6	Tourist	Welcome I Say	2023-07-13 19:48:10	5
7	jiangly	Welcome I Say	2023-06-08 03:10:56	5
8	Benq	Welcome I Say	2023-05-11 12:37:42	6
9	HussBak	you guys did an am	2023-04-04 06:25:33	3
10	ImanGha	you guys did an am	2023-03-01 23:50:20	2

Admin

username	role	contributionScore
HussBak	superAdmin	2394
ImanGha	moderator	1001
Khaled47	supportAdmin	1000

Responsibilites

adminUsername	Responsibility
HussBak	Assesses Contest Creators
ImanGha	User Management
Khaled47	Technical Support

Notifies

creatorUsername	contestantUsername	problemID	notificationContent
HussBak	Khaled47	2	There is an error. We meant that exactly one ball should be taken out not atleast
HussBak	AliFarhat	2	There is an error. We meant that exactly one ball should be taken out not atleast
HussBak	ay_sharaf	2	There is an error. We meant that exactly one ball should be taken out not atleast
ImanGha	Dankosay	5	Read the problem statement well. Atmost one potato should be eaten
ImanGha	Tourist	5	Read the problem statement well. Atmost one potato should be eaten

ContestCreator

creatorUsername	assessmentScore
HussBak	2445
ImanGha	2543

Organizes

creatorUsername	roundNumber
HussBak	1
ImanGha	2
HussBak	3
ImanGha	4
HussBak	5
ImanGha	6
HussBak	7
ImanGha	8
HussBak	9
ImanGha	10

Submits

usernames	submissionID	problemID
Khaled47	1	2
Dankosay	1	4
Khaled47	2	8
ay_sharaf	2	1
Khaled47	3	2
HussBak	3	4
HussBak	4	6

Hacks

hackerUsername	submissionID	hackID
ay_sharaf	1	1
Khaled47	2	2
Dankosay	3	3

CompetesIn

contestantUsername	roundNumber	score	rank	timestampOfRegistration
Tourist	2	1000	56	1/1/2024-1:00:00
jiangly	3	800	34	1/2/2024-1:00:00
Benq	3	900	23	1/3/2024-1:00:00
ay_sharaf	1	700	4355	1/4/2024-1:00:00
AliFarhat	1	800	1232	1/5/2024-1:00:00
Khaled47	1	800	4532	1/6/2024-1:00:00
Khaled47	4	500	2342	1/7/2024-1:00:00
Dankosay	2	0	8532	1/8/2024-1:00:00

hack

hackID	test	defender	verdict	timestamp
1	65	Khaled47	fail	1/1/2024-5:00:00
2	35	ay_sharaf	fail	1/2/2024-6:00:00
3	24	AliFarhat	fail	1/3/2024-6:30:00

Solution

problemID	SolutionID	Explanation	SourceCode
1	1	Greedy	#include <iostream>
2	2	Dynamic Programming	#include <iostream>
3	3	Two Pointers	#include <iostream>
4	4	Divide and Conquer	#!/usr/bin/env python3
5	5	Heap Data Structure	#include <iostream>
6	6	Set Data Structure	#include <iostream>
7	7	queue Data Structure	#include <iostream>
8	8	Graph Traversal	#include <iostream>
9	9	Fast Fourier Transform	public class AddOneToTwo {
10	10	Combinatorics	import random
11	11	Greedy	#include <iostream>
12	12	Dynamic Programming	#include <iostream>
13	13	Two Pointers	#include <iostream>
14	14	Divide and Conquer	#include <iostream>
15	15	Heap Data Structure	#!/usr/bin/env python3
16	16	Set Data Structure	#include <iostream>
17	17	queue Data Structure	#include <iostream>
18	18	Graph Traversal	#include <iostream>
19	19	Fast Fourier Transform	#include <iostream>
20	20	Combinatorics	public class AddOneToTwo {
21	21	Greedy	import random
22	22	Dynamic Programming	#include <iostream>
23	23	Two Pointers	#include <iostream>
24	24	Divide and Conquer	#include <iostream>
25	25	Heap Data Structure	#include <iostream>
26	26	Set Data Structure	#include <iostream>
27	27	queue Data Structure	#!/usr/bin/env python3
28	28	Graph Traversal	#include <iostream>
29	29	Fast Fourier Transform	#include <iostream>
30	30	Combinatorics	#include <iostream>

11- Building the Database via SQL Queries

11.1-Creating the Tables

The following section is for creating the tables in pgAdmin 4 via postgresql.

11.1.1- User:

```
CREATE TABLE IF NOT EXISTS public."User"
(
    username text COLLATE pg_catalog."default" NOT NULL,
    email text COLLATE pg_catalog."default",
    passwod text COLLATE pg_catalog."default",
    "firstName" text COLLATE pg_catalog."default",
    "lastName" text COLLATE pg_catalog."default",
    country text COLLATE pg_catalog."default",
    "registrationDate" date,
    CONSTRAINT "User_pkey" PRIMARY KEY (username)
)
```

Chosen Datatypes for the attributes:

- username: text
- email: text
- firstName: text
- lastName: text
- country: text

Constraints:

- not NULL: username
- primary key: username

- foreign key(s): none

11.1.2- Messages:

```
CREATE TABLE IF NOT EXISTS public."Messages"
(
    "senderUsername" text COLLATE pg_catalog."default" NOT NULL,
    "receiverUsername" text COLLATE pg_catalog."default" NOT NULL,
    content text COLLATE pg_catalog."default",
    "timestamp" timestamp without time zone NOT NULL,
    CONSTRAINT "Messages_pkey" PRIMARY KEY ("senderUsername", "receiverUsername", "timestamp"),
    CONSTRAINT user2_fk FOREIGN KEY ("receiverUsername")
        REFERENCES public."User" (username) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID,
    CONSTRAINT user_fk FOREIGN KEY ("senderUsername")
        REFERENCES public."User" (username) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)
```

Chosen Datatypes for the attributes:

- senderUsername: text
- receiverUsername: text
- content: text

- timestamp: timestamp without time zone

Constraints:

- not NULL: senderUsername, receiverUsername, timestamp
- primary key: senderUsername, receiverUsername, timestamp
- foreign key(s):
    1- senderUsername references username in User table
    referential triggered action on Update/Delete: Cascade (propagate the effect)
    2- receiverUsername references username in User table
    referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.3- Befriends:

```
CREATE TABLE IF NOT EXISTS public."Befriends"
(
    "friend1Username" text COLLATE pg_catalog."default" NOT NULL,
    "friend2Username" text COLLATE pg_catalog."default" NOT NULL,
    "timestamp" timestamp without time zone,
    CONSTRAINT "Befriends_pkey" PRIMARY KEY ("friend1Username", "friend2Username"),
    CONSTRAINT friend1_fk FOREIGN KEY ("friend1Username")
        REFERENCES public."User" (username) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT friend2_fk FOREIGN KEY ("friend2Username")
        REFERENCES public."User" (username) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
```

```
)
```

Chosen Datatypes for the attributes:

- friend1Username: text
- friend2Username: text
- timestamp: timestamp without time zone

Constraints:

- not NULL: friend1Username, friend2Username, timestamp
- primary key: friend1Username, friend2Username, timestamp
- foreign key(s):
  1- friend1Username references username in User table
  referential triggered action on Update/Delete: Cascade (propagate the effect)
  2- friend2Username references username in User table
  referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.4- BlogEntry:

```
CREATE TABLE IF NOT EXISTS public."BlogEntry"
(
    "blogEntryID" serial NOT NULL,
    username text COLLATE pg_catalog."default",
    title text COLLATE pg_catalog."default",
    content text COLLATE pg_catalog."default",
    "timestamp" timestamp without time zone,
    CONSTRAINT "BlogEntry_pkey" PRIMARY KEY ("blogEntryID"),
    CONSTRAINT user_fk FOREIGN KEY (username)
        REFERENCES public."User" (username) MATCH SIMPLE
        ON UPDATE CASCADE
```

```
    ON DELETE CASCADE
)
```

Chosen Datatypes for the attributes:

- blogEntryID: serial (autoincrement integer)
- username: text
- title: text
- content: text
- timestamp: timestamp without time zone

Constraints:

- not NULL: blogEntryID
- primary key: blogEntryID
- foreign key(s):
    1- username references username in User table
    referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.5- Hashtag:

```
CREATE TABLE IF NOT EXISTS public."Hashtag"
(
  "blogEntryID" integer NOT NULL,
  tag text COLLATE pg_catalog."default" NOT NULL,
  CONSTRAINT "Hashtag_pkey" PRIMARY KEY ("blogEntryID", tag),
  CONSTRAINT "blogEntry_fk" FOREIGN KEY ("blogEntryID")
    REFERENCES public."BlogEntry" ("blogEntryID") MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE
)
```

Chosen Datatypes for the attributes:

- blogEntryID: integer
- tag: text

Constraints:

- not NULL: blogEntryID, tag
- primary key: blogEntryID, tag
- foreign key(s):
  1- blogEntryID references blogEntryID in BlogEntry table
  referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.6 - Comments:

```
CREATE TABLE IF NOT EXISTS public."Comments"
(
    "blogEntryID" integer NOT NULL,
    username text COLLATE pg_catalog."default" NOT NULL,
    content text COLLATE pg_catalog."default",
    "timestamp" timestamp without time zone NOT NULL,
    CONSTRAINT "Comments_pkey" PRIMARY KEY ("blogEntryID", username, "timestamp"),
    CONSTRAINT "blogEntry_fk" FOREIGN KEY ("blogEntryID")
        REFERENCES public."BlogEntry" ("blogEntryID") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT user_fk FOREIGN KEY (username)
        REFERENCES public."User" (username) MATCH SIMPLE
        ON UPDATE CASCADE
```

```
    ON DELETE CASCADE
)
```

Chosen Datatypes for the attributes:

- blogEntryID: integer
- username: text
- content: text
-timestamp: timestamp without timezone

Constraints:

- not NULL: blogEntryID, username, timestamp
- primary key: blogEntryID, username, timestamp (timestamp so that a user can comment on a blog entry several times)
- foreign key(s):
    1- blogEntryID references blogEntryID in BlogEntry table
    referential triggered action on Update/Delete: Cascade (propagate the effect)
    2- username references username in User table
    referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.7 - VotesOn:

```
CREATE TABLE IF NOT EXISTS public."VotesOn"
(
    "blogEntryID" integer NOT NULL,
    username text COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Votes_On_pkey" PRIMARY KEY ("blogEntryID", username),
    CONSTRAINT blog_fk FOREIGN KEY ("blogEntryID")
        REFERENCES public."BlogEntry" ("blogEntryID") MATCH SIMPLE
        ON UPDATE CASCADE
```

```
    ON DELETE CASCADE,
  CONSTRAINT user_fk FOREIGN KEY (username)
    REFERENCES public."User" (username) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE
    NOT VALID
)
```

Chosen Datatypes for the attributes:

- blogEntryID: integer
- username: text

Constraints:

- not NULL: blogEntryID, username
- primary key: blogEntryID, username(no timestamp so that a user can vote on a blog entry only once)
- foreign key(s):
  1- blogEntryID references blogEntryID in BlogEntry table
  referential triggered action on Update/Delete: Cascade (propagate the effect)
  2- username references username in User table
  referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.8 - Announcement:

```
CREATE TABLE IF NOT EXISTS public."Announcement"
(
  "announcementID" serial NOT NULL,
  username text COLLATE pg_catalog."default",
  language text COLLATE pg_catalog."default",
  content text COLLATE pg_catalog."default",
```

```
    "timestamp" timestamp without time zone,
    CONSTRAINT "Announcement_pkey" PRIMARY KEY ("announcementID"),
    CONSTRAINT user_fk FOREIGN KEY (username)
        REFERENCES public."User" (username) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)
```

Chosen Datatypes for the attributes:

- announcementID: serial (autoincrement integer)
- username: text
- language: text
- content: text
- timestamp: timestamp without timezone

Constraints:

- not NULL: announcementID
- primary key: announcementID
- foreign key(s):
    1- username references username in User table
    referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.9 - Admin:

```
CREATE TABLE IF NOT EXISTS public."Admin"
(
    username text COLLATE pg_catalog."default" NOT NULL,
    role text COLLATE pg_catalog."default",
```

```
    "contributionScore" integer,
    CONSTRAINT "Admin_pkey" PRIMARY KEY (username),
    CONSTRAINT user_fk FOREIGN KEY (username)
        REFERENCES public."User" (username) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)
```

Chosen Datatypes for the attributes:

- username: text
- role: text
- contributionScore: integer

Constraints:

- not NULL: username
- primary key: username
- foreign key(s):
    1- username references username in User table
    referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.10 - Responsibilities:

```
CREATE TABLE IF NOT EXISTS public."Responsibilities"
(
    "adminUsername" text COLLATE pg_catalog."default" NOT NULL,
    responsibility text COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Responsibilities_pkey" PRIMARY KEY ("adminUsername", responsibility),
    CONSTRAINT admin_fk FOREIGN KEY ("adminUsername")
        REFERENCES public."Admin" (username) MATCH SIMPLE
```

```
    ON UPDATE CASCADE
    ON DELETE CASCADE
)
```

Chosen Datatypes for the attributes:

- adminUsername: text
- responsibility: text

Constraints:

- not NULL: adminUsername, responsibility
- primary key: adminUsername, responsibility
- foreign key(s):
    1- adminUsername references username in User table
    referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.11 - ContestCreator:

```
CREATE TABLE IF NOT EXISTS public."ContestCreator"
(
  "creatorUsername" text COLLATE pg_catalog."default" NOT NULL,
  "assessmentScore" integer,
  CONSTRAINT "ContestCreator_pkey" PRIMARY KEY ("creatorUsername"),
  CONSTRAINT user_fk FOREIGN KEY ("creatorUsername")
    REFERENCES public."User" (username) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE
)
```

Chosen Datatypes for the attributes:

- creatorUsername: text
- assessmentScore: integer

Constraints:

- not NULL: creatorUsername
- primary key: creatorUsername
- foreign key(s):
    1- creatorUsername references username in User table
    referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.12 - Contestant:

```
CREATE TABLE IF NOT EXISTS public."Contestant"
(
    "contestantUsername" text COLLATE pg_catalog."default" NOT NULL,
    rating integer,
    CONSTRAINT "Contestant_pkey" PRIMARY KEY ("contestantUsername"),
    CONSTRAINT user_fk FOREIGN KEY ("contestantUsername")
        REFERENCES public."User" (username) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)
```

Chosen Datatypes for the attributes:

- contestantUsername: text
- rating: integer

Constraints:

- not NULL: contestantUsername
- primary key: contestantUsername
- foreign key(s):
    1- contestantUsername references username in User table
       referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.13 - Contest:

```
CREATE TABLE IF NOT EXISTS public."Contest"
(
    "roundNumber" serial NOT NULL,
    name text COLLATE pg_catalog."default",
    divsion integer,
    "startTimestamp" timestamp without time zone,
    length time without time zone,
    description text COLLATE pg_catalog."default",
    CONSTRAINT "Contest_pkey" PRIMARY KEY ("roundNumber")
)
```

Chosen Datatypes for the attributes:
- roundNumber: serial (autoincrement integer)
- name: text
-division: integer
-startTimestamp: timestamp without time zone
-length: time without time zone
- description: text

Constraints:
- not NULL: roundNumber

- primary key: roundNumber
- foreign key(s): none

11.1.14 - Organizes:

```
CREATE TABLE IF NOT EXISTS public."Organizes"
(
    "creatorUsername" text COLLATE pg_catalog."default" NOT NULL,
    "roundNumber" integer NOT NULL,
    CONSTRAINT "Organizes_pkey" PRIMARY KEY ("creatorUsername", "roundNumber"),
    CONSTRAINT contest_fk FOREIGN KEY ("roundNumber")
        REFERENCES public."Contest" ("roundNumber") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID,
    CONSTRAINT creator_fk FOREIGN KEY ("creatorUsername")
        REFERENCES public."ContestCreator" ("creatorUsername") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)
```

Chosen Datatypes for the attributes:

- creatorUsername: text
- roundNumber: integer

Constraints:

- not NULL: creatorUsername, roundNumber

- primary key: creatorUsername, roundNumber
- foreign key(s):
    1- creatorUsername references username in User table
       referential triggered action on Update/Delete: Cascade (propagate the effect)
    2- roundNumber references roundNumber in Contest table
       referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.15 - CompetesIn:

```
CREATE TABLE IF NOT EXISTS public."CompetesIn"
(
    "contestantUsername" text COLLATE pg_catalog."default" NOT NULL,
    "roundNumber" integer NOT NULL,
    score integer,
    rank integer,
    "timestampOfRegistration" timestamp without time zone,
    CONSTRAINT "CompetesIn_pkey" PRIMARY KEY ("contestantUsername", "roundNumber"),
    CONSTRAINT contest_fk FOREIGN KEY ("roundNumber")
        REFERENCES public."Contest" ("roundNumber") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID,
    CONSTRAINT contestant_fk FOREIGN KEY ("contestantUsername")
        REFERENCES public."Contestant" ("contestantUsername") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
```

```
)
```

Chosen Datatypes for the attributes:

- contestantUsername: text
- roundNumber: integer
- score: integer
- rank: integer
- timestampOfRegistration: timestamp without time zone,

Constraints:

- not NULL: contestantUsername, roundNumber
- primary key: contestantUsername, roundNumber
- foreign key(s):
  1- contestantUsername references username in User table
  referential triggered action on Update/Delete: Cascade (propagate the effect)
  2- roundNumber references roundNumber in Contest table
  referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.16 - ProgrammingProblem:

```
CREATE TABLE IF NOT EXISTS public."ProgrammingProblem"
(
    "problemID" serial NOT NULL,
    "difficultyLevel" character(1) COLLATE pg_catalog."default",
    title text COLLATE pg_catalog."default",
    description text COLLATE pg_catalog."default",
    "timeLimit" integer,
    "memoryLimit" integer,
    "roundNumber" integer,
    CONSTRAINT "ProgrammingProblem_pkey" PRIMARY KEY ("problemID"),
```

```
    CONSTRAINT contest_fk FOREIGN KEY ("roundNumber")
        REFERENCES public."Contest" ("roundNumber") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)
```

Chosen Datatypes for the attributes:

- problemID: serial (autoincrement integer)
- difficultyLevel: character(1)
-title: text
-description: text
- timeLimit: integer
- memoryLimit: integer
- roundNumber: integer

Constraints:

- not NULL: problemID
- primary key: problemID
- foreign key(s):
    1- roundNumber references roundNumber in Contest table
    referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.17 - Tag:

```
CREATE TABLE IF NOT EXISTS public."Tag"
(
    "problemID" integer NOT NULL,
    tag text COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Tag_pkey" PRIMARY KEY ("problemID", tag),
```

```
    CONSTRAINT problem_fk FOREIGN KEY ("problemID")
        REFERENCES public."ProgrammingProblem" ("problemID") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)
```

Chosen Datatypes for the attributes:

- problemID: integer
- tag: text

Constraints:

- not NULL: problemID, tag
- primary key: problemID, tag
- foreign key(s):
    1- problemID references problemID in ProgrammingProblem table
    referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.18 - TestCase:

```
CREATE TABLE IF NOT EXISTS public."TestCase"
(
    "problemID" integer NOT NULL,
    "testCaseNumber" serial NOT NULL,
    input text COLLATE pg_catalog."default",
    "expectedOutput" text COLLATE pg_catalog."default",
    CONSTRAINT "TestCase_pkey" PRIMARY KEY ("problemID", "testCaseNumber"),
    CONSTRAINT problem_fk FOREIGN KEY ("problemID")
        REFERENCES public."ProgrammingProblem" ("problemID") MATCH SIMPLE
```

```
    ON UPDATE CASCADE

    ON DELETE CASCADE

)
```

Chosen Datatypes for the attributes:

- problemID: integer
- testCaseNumber: serial (autoincrement integer)
- input: text

Constraints:

- not NULL: problemID, testCaseNumber
- primary key: problemID, testCaseNumber
- foreign key(s):
    1- problemID references problemID in ProgrammingProblem table
    referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.19 - Solution:

```
CREATE TABLE IF NOT EXISTS public."Solution"

(

  "problemID" integer NOT NULL,

  "solutionID" serial NOT NULL,

  "sourceCode" text COLLATE pg_catalog."default",

  CONSTRAINT "Solution_pkey" PRIMARY KEY ("solutionID", "problemID"),

  CONSTRAINT problem_fk FOREIGN KEY ("problemID")

    REFERENCES public."ProgrammingProblem" ("problemID") MATCH SIMPLE

    ON UPDATE CASCADE

    ON DELETE CASCADE
```

```
)
```

Chosen Datatypes for the attributes:

- problemID: integer
- solutionID: serial (autoincrement integer)
- sourceCode: text

Constraints:

- not NULL: problemID, solutionID
- primary key: problemID, solutionID
- foreign key(s):
  1- problemID references problemID in ProgrammingProblem table
  referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.20 - AddsNotification:

```
CREATE TABLE IF NOT EXISTS public."AddsNotification"
(
    "creatorUsername" text COLLATE pg_catalog."default" NOT NULL,
    "roundNumber" integer NOT NULL,
    "timestamp" timestamp without time zone NOT NULL,
    "notificationContent" text COLLATE pg_catalog."default",
    CONSTRAINT "AddsNotification_pkey" PRIMARY KEY ("timestamp", "roundNumber",
"creatorUsername"),
    CONSTRAINT contest_fk FOREIGN KEY ("roundNumber")
        REFERENCES public."Contest" ("roundNumber") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
```

```
    NOT VALID,
  CONSTRAINT creator_fk FOREIGN KEY ("creatorUsername")
    REFERENCES public."ContestCreator" ("creatorUsername") MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE
    NOT VALID
)
```

Chosen Datatypes for the attributes:

- creatorUsername: text
- roundNumber: integer
- timestamp: timestamp without timezone
- notificationContent: text

Constraints:

- not NULL: creatorUsername, roundNumber, timestamp
- primary key: creatorUsername, roundNumber, timestamp (timestamp so that a contest creator can add a notification to a contest several times)

- foreign key(s):
  1- creatorUsername references creatorUsername in ContestCreator table
  referential triggered action on Update/Delete: Cascade (propagate the effect)
  2- roundNumber references roundNumber in Contest table
  referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.21 - Submission:

```
CREATE TABLE IF NOT EXISTS public."Submission"
(
  "submissionID" serial NOT NULL,
```

```
    "programmingLanguage" text COLLATE pg_catalog."default",
    "sourceCode" text COLLATE pg_catalog."default",
    verdict text COLLATE pg_catalog."default",
    "timestamp" timestamp without time zone,
    CONSTRAINT "Submission_pkey" PRIMARY KEY ("submissionID")
)
```

Chosen Datatypes for the attributes:

- submissionID: serial (autoincrement integer)
- programmingLanguage: text
- sourceCode: text
- verdict: text
- timestamp: timestamp without timezone

Constraints:

- not NULL: submissionID
- primary key: submissionID
- foreign key(s): none

11.1.22 - Submits:

```
CREATE TABLE IF NOT EXISTS public."Submits"
(
    username text COLLATE pg_catalog."default" NOT NULL,
    "submissionID" integer NOT NULL,
    "problemNumber" integer NOT NULL,
    CONSTRAINT "Submits_pkey" PRIMARY KEY ("problemNumber", "submissionID", username),
    CONSTRAINT problem_fk FOREIGN KEY ("problemNumber")
```

```
    REFERENCES public."ProgrammingProblem" ("problemID") MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE
    NOT VALID,
  CONSTRAINT submission_fk FOREIGN KEY ("submissionID")
    REFERENCES public."Submission" ("submissionID") MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE
    NOT VALID,
  CONSTRAINT user_fk FOREIGN KEY (username)
    REFERENCES public."User" (username) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE
)
```

Chosen Datatypes for the attributes:

- username: text
- submissionID: integer
- problemNumber: integer

Constraints:

- not NULL: username, submissionID, problemNumber
- primary key: username, submissionID, problemNumber
- foreign key(s):
  1- username references username in User table
  referential triggered action on Update/Delete: Cascade (propagate the effect)
  2- submissionID references submissionID in Submission table
  referential triggered action on Update/Delete: Cascade (propagate the effect)

3- problemNumber references problemID in ProgrammingProblem table
referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.23 - Hack:

```
CREATE TABLE IF NOT EXISTS public."Hack"
(
    "hackID" serial NOT NULL,
    test text COLLATE pg_catalog."default",
    verdict text COLLATE pg_catalog."default",
    "timestamp" timestamp without time zone,
    CONSTRAINT "Hack_pkey" PRIMARY KEY ("hackID")
)
```

Chosen Datatypes for the attributes:

- hackID: serial (autoincrement integer)
- test: text
- verdict: text
- timestamp: timestamp without time zone

Constraints:

- not NULL: hackID
- primary key: hackID
- foreign key(s): none

11.1.24 - Hacks:

```
CREATE TABLE IF NOT EXISTS public."Hacks"
(
    "hackerUsername" text COLLATE pg_catalog."default" NOT NULL,
```

```
    "submissionID" integer NOT NULL,
    "hackID" integer NOT NULL,
    CONSTRAINT "Hacks_pkey" PRIMARY KEY ("hackerUsername", "submissionID", "hackID"),
    CONSTRAINT hack_fk FOREIGN KEY ("hackID")
        REFERENCES public."Hack" ("hackID") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID,
    CONSTRAINT hacker_fk FOREIGN KEY ("hackerUsername")
        REFERENCES public."Contestant" ("contestantUsername") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT submission_fk FOREIGN KEY ("submissionID")
        REFERENCES public."Submission" ("submissionID") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
)
```

Chosen Datatypes for the attributes:

- hackerUsername: text
- submissionID: integer
- hackID: integer

Constraints:

- not NULL: hackerUsername, submissionID, hackID

- primary key: hackerUsername, submissionID, hackID
- foreign key(s):
    1- hackerUsername references contestantUsername in Contestant table
    referential triggered action on Update/Delete: Cascade (propagate the effect)
    2- submissionID references submissionID in Submission table
    referential triggered action on Update/Delete: Cascade (propagate the effect)
    3- hackID references hackID in Hack table
    referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.25 - Battle:

```
CREATE TABLE IF NOT EXISTS public."Battle"

(

    "battleID" serial NOT NULL,

    "startTimestamp" timestamp without time zone,

    "endTimestamp" timestamp without time zone,

    CONSTRAINT "Battle_pkey" PRIMARY KEY ("battleID")

)
```

Chosen Datatypes for the attributes:

- battleID: serial (autoincrement integer)
- startTimestamp: timestamp without time zone,
- endTimestamp” timestamp without time zone,

Constraints:

- not NULL: battleID
- primary key: battleID
- foreign key(s): none

11.1.26 - Joins:

```
CREATE TABLE IF NOT EXISTS public."Joins"

(
```

```
    username text COLLATE pg_catalog."default" NOT NULL,
    "battleID" integer NOT NULL,
    "isWinner" boolean,
    CONSTRAINT "Joins_pkey" PRIMARY KEY ("battleID", username),
    CONSTRAINT battle_fk FOREIGN KEY ("battleID")
        REFERENCES public."Battle" ("battleID") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT user_fk FOREIGN KEY (username)
        REFERENCES public."User" (username) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)
```

Chosen Datatypes for the attributes:

- username: text
- battleID: integer
- isWinner: boolean

Constraints:

- not NULL: battleID, username
- primary key: battleID, username
- foreign key(s):
    1- username references username in User table
    referential triggered action on Update/Delete: Cascade (propagate the effect)
    2- battleID references battleID in Battle table
    referential triggered action on Update/Delete: Cascade (propagate the effect)

11.1.27 - ConsistsOf:

```
CREATE TABLE IF NOT EXISTS public."ConsistsOf"

(

   "battleID" integer NOT NULL,

   "problemID" integer NOT NULL,

   CONSTRAINT "ConsistsOf_pkey" PRIMARY KEY ("battleID", "problemID"),

   CONSTRAINT battle_fk FOREIGN KEY ("battleID")

      REFERENCES public."Battle" ("battleID") MATCH SIMPLE

      ON UPDATE CASCADE

      ON DELETE CASCADE,

   CONSTRAINT problem_fk FOREIGN KEY ("problemID")

      REFERENCES public."ProgrammingProblem" ("problemID") MATCH SIMPLE

      ON UPDATE CASCADE

      ON DELETE CASCADE

      NOT VALID

)
```

Chosen Datatypes for the attributes:

- battleID: integer
- problemID: integer

Constraints:

- not NULL: battleID, problemID
- primary key: battleID, problemID
- foreign key(s):
    1- battleID references battleID in Battle table

referential triggered action on Update/Delete: Cascade (propagate the effect)

2- problemID references problemID in ProgrammingProblem table

referential triggered action on Update/Delete: Cascade (propagate the effect)

11.2- Checks, Triggers & Constraints

The following section is for checks, triggers, and constraints used to force logical and value constraints on the domains.

11.2.1- Check for Division in Contest:

The domain for a contest's division is {1,2,3,4}, 4 being the easiest and 1 being the toughest.

```
ALTER TABLE public."Contest"

ADD CONSTRAINT check_division CHECK ("divsion"=1 or "divsion"=2 or "divsion"=3 or "divsion"=4);
```

11.2.2- Check for Difficulty Level in ProgrammingProblem:

The domain for a programming problem's difficulty level is {A, B, C, D, E, F}, A being the easiest and F being the toughest.

```
ALTER TABLE public."ProgrammingProblem"

ADD CONSTRAINT check_difficultyLevel CHECK ("difficultyLevel"='A' or "difficultyLevel"='B' or
"difficultyLevel"='C' or "difficultyLevel"='D' or "difficultyLevel"='E' or "difficultyLevel"='F' );
```

11.2.3- Check for Verdict in Submission:

The domain for a submission's verdict is {Accepted, Wrong Answer , Hacked}

```
ALTER TABLE public."Submission"

ADD CONSTRAINT check_verdict CHECK ("verdict"='Accepted' or "verdict"='Wrong Answer' or
"verdict"='Hacked');
```

11.2.4- Check for Programming Language in Submission:

The domain for a submission's programming language is {GNU G++20 11.2.0, GNU G++17 7.3.0, GNU G++14 6.4.0, Python 2.7.18, Python 3.8.10, Java 11.0.6, Java 17 64bit, Java 21 64bit, OCaml 4.02.1, C# 8, C# 10}

```
ALTER Table public."Submission"

ADD CONSTRAINT check_problem_language CHECK ("programmingLanguage" = 'GNU G++20 11.2.0' or "programmingLanguage" = 'GNU G++17 7.3.0' or "programmingLanguage" = 'GNU G++14 6.4.0' or "programmingLanguage" ='Python 2.7.18' or "programmingLanguage" ='Python 3.8.10' or "programmingLanguage" ='Java 11.0.6' or "programmingLanguage" ='Java 17 64bit' or "programmingLanguage" = 'Java 21 64bit' or "programmingLanguage" ='OCaml 4.02.1' or "programmingLanguage" ='C# 8' or "programmingLanguage" ='C# 10');
```

11.2.5- Check for Language in Announcement:

The domain for an announcement's language is {English, Arabic, French, German, Italian, Spanish, Russian}

```
ALTER Table public."Announcement"

ADD CONSTRAINT check_announcement_language CHECK ("language" = 'English' or "language" = 'Arabic' or "language" = 'French' or "language" = 'German' or "language" = 'Italian' or "language" = 'Spanish' or "language" = 'Russian');
```

11.2.6- Check for Non-Negative Rating in Contestant

A contestant must always have a non-negative rating.

```
ALTER TABLE public."Contestant"

ADD CONSTRAINT check_positive_rating check ("rating" >= 0);
```

11.2.7- Check for Positive Rank in Contestant

A contestant must always have a positive rank when competing in a contest.

```
ALTER TABLE public."CompetesIn"

ADD CONSTRAINT check_positive_rank check ("rank" > 0);
```

11.2.8- Check for Positive Limits in Programming Problem

A programming problem must have a positive time limit and a positive memory limit.

```
ALTER TABLE public."ProgrammingProblem"

ADD CONSTRAINT check_positive_limits check ("timeLimit" >0 and "memoryLimit" >0);
```

11.2.9- Check for Valid Email in User

A user's email must be of the form reg@reg.reg. That's why a regular expression was used to ensure that the email is in a valid format.

```
ALTER Table public."User"

ADD CONSTRAINT check_valid_email CHECK(REGEXP_LIKE("email", '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-
]+\.[A-Za-z]{2,}$'));
```

11.2.10- Check for 2 Participants Only Per Battle in Joins

Battles are designed so that only 2 users can join them.

```
CREATE OR REPLACE FUNCTION check_2_participants()

RETURNS TRIGGER AS $$

BEGIN

        IF(SELECT Count(*) FROM public."Joins" WHERE "battleID" = NEW. "battleID")>1 THEN
```

```
            RETURN NULL;
        End IF;
        Return NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_two_participants_per_battle
Before insert on public."Joins"
FOR EACH ROW EXECUTE FUNCTION check_2_participants();
```

11.2.11- Check for 2-3 Problems Only Per Battle in ConsistOf

Battles are designed so that they can consist of only 2 or 3 problems.

```
CREATE OR REPLACE FUNCTION check_problems_per_battles()
RETURNS TRIGGER AS $$
BEGIN
        IF(SELECT Count(*) FROM public."ConsistsOf" WHERE "battleID" = NEW. "battleID")>2 THEN
            RETURN NULL;
        End IF;
        Return NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_two_or_three_problems_per_battle
```

```
Before insert on public."ConsistsOf"

FOR EACH ROW EXECUTE FUNCTION check_problems_per_battles();
```

11.2.12- Constraint for Unique Submission ID in Submits

A submission can only be submitted once.

```
ALTER TABLE public."Submits"

ADD CONSTRAINT unique_submission_ID_constraint UNIQUE ("submissionID");
```

11.2.13- Check for Eligibility in CompetesIn:

In order to even out the playing field, requirements for competing in contests are put in place to ensure fairness when it comes to the rating of the competitors. Contestants with very high ratings (>1900) are not allowed to participate in division 3 or 4 contests which are on the easier side. This aims to give a chance to lower-level players to attain a good ranking. Similarly, contestants with lower ratings (<1600) are not allowed to participate in the rigorous, division 4 contests.

```
CREATE OR REPLACE FUNCTION check_contest_eligibility()

RETURNS TRIGGER AS $$

DECLARE

   contestant_rating integer;

   contest_division integer;

BEGIN

        SELECT "rating"

   INTO contestant_rating

   FROM public."Contestant" co
```

```
    WHERE co."contestantUsername" = NEW."contestantUsername";


            SELECT "divsion"
    INTO contest_division
    FROM public."Contest" con
    WHERE con."roundNumber" = NEW."roundNumber";


            IF ((contestant_rating > 1900 AND contest_division <= 2)
        OR
        (contestant_rating <= 1900 AND contestant_rating >= 1600 AND contest_division<= 3 AND
contest_division > 1)
        OR
        (contestant_rating < 1600 AND contest_division > 1)
    ) THEN
                        RETURN NEW;
            End IF;
            Return NULL;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER check_contest_eligibility_trigger
Before insert on public."CompetesIn"
FOR EACH ROW EXECUTE FUNCTION check_contest_eligibility();
```

11.3-Inserting Data into Tables

The following section is for inserting the data into the database tables.

11.3.1- User:

```
INSERT INTO public."User"(
    username, email, passwod, "firstName", "lastName", country, "registrationDate")
VALUES
    ('Khaled47', 'Khaled47@gmail.com',
'5e884898da28047151d0e56f8dc6292773603d0d6b8b4b9b883883a1541d3343', 'Khaled', 'Charaf',
'Lebanon', '2023-01-15'),
    ('AliFarhat', 'AliFarhat@gmail.com',
'2d3e1312b4f10a8a7c0ebc7dc5ea8cf2cbca99e7c8bd6a57f188e7e1e5571d7', 'Ali', 'Farhat', 'Lebanon',
'2022-10-21'),
    ('ay_charaf', 'ay_charaf@gmail.com',
'3a63c03f43710ad8eae4c0d6de315df0d73f0b3b5f5ae8e09d4da598b968c66', 'Ayman', 'Charaf',
'Lebanon', '2023-06-08'),
    ('HussKiller', 'HussKiller@gmail.com',
'ad8b3b7d8755165555a80f8b0b5f196ea44d6c0a54f491b7a2a66e8dfdece0b', 'Hussein', 'Bakri',
'Lebanon', '2022-07-30'),
    ('ImanDestroyer', 'ImanDestroyer@gmail.com',
'ea4109f401e9aabbab4c5fb17c25f31e5c64bfa5e1efcdfcf4c9ec9910b896d', 'Iman', 'Ghalayini',
'Lebanon', '2023-03-04'),
    ('Danosour', 'Danosour@gmail.com',
'1f548a2d8992b4d8a6cb424f3c6aeef19e719e5d158c6fb39e9ea2ae23f4570', 'Dana', 'Kossaybeyati',
'Lebanon', '2022-08-16'),
```

('HashTrash', 'HashTrash@gmail.com', 'f5297c5e46c2f95e5c700f946e77dd715583cfb41895fcb52e4d85b7eb38127', 'Hashem', 'Khodor', 'Lebanon', '2023-02-14'),

('TheJoker', 'TheJoker@gmail.com', 'efbc56e0d798f9372831ceab35f8c3d812ead0c4e1c1ac21f96c0f8bb4249f7', 'Omar', 'Ramadan', 'Lebanon', '2023-02-04'),

('MommyBoy', 'MommyBoy@gmail.com', 'fd1fc301decbde0a5ad5c4c27ae398d07bb9ec75c7e2c1b11a76d0068d15cd', 'Abdelatif', 'Saleh', 'Lebanon', '2023-01-29'),

('cutiepie', 'cutiepie@gmail.com', 'e2bf8da1bde8ed5e6ccf3f936da984ae11a2c0403c54f64eb69a1e7b1d3d62c', 'Lama', 'Hasbini', 'America', '2023-02-21'),

('Tourist', 'Tourist@gmail.com', '5e884898da28047151d0e56f8dc6292773603d0d6b8b4b9b883883a1541d3343', 'Gennady', 'Korotkevich', 'Belarus', '2022-08-16'),

('BlueCoder42', 'bluecoder@gmail.com', 'YSBN93u28y382', 'Alex', 'Turner', 'Japan', '2023-04-15'),

('TechNinja91', 'techtech@mail.com', 'sUb292372804', 'Maya', 'Rodriguez', 'Brazil', '2022-09-28'),

('QuantumCoder7', 'quantumcoder@mail.com', '482t40', 'Ethan', 'Bennett', 'Australia', '2023-07-03'),

('CodeMasterX', 'masterrr23@gmail.com', 'suhsYV27', 'Olivia', 'Chang', 'Germany', '2022-11-22'),

('PixelGeek23', 'pixy77@mail.com', '826HVGT2', 'Liam', 'Mitchell', 'India', '2020-05-18'),

('CyberPioneer', 'cyber101@gmail.com', 'dhYGF27', 'Ava', 'Thompson', 'Canada', '2020-08-09'),

('SwiftSorcerer', 'swiftt1@mail.com', '7S28B2G', 'Noah', 'Patel', 'South Africa', '2022-12-30'),

('BinaryExplorer88', 'binbin77@mail.com', 'SSS8oj1b', 'Mia', 'Sanchez','Lebanon', '2019-01-10'),

('MK', 'MK@gmail.com','6600bf6331c98f1619619fc8e405d5ec3dd97e614b9d241f14275168f130c916', 'Mohamad', 'Kreidieh', 'Lebanon', current_timestamp),

('TamTam',
'TamTam@gmail.com','6c3bf86f56a3e67d88f6d6bea7e03dd5ca3c31b07a51fc1ce28571e049dce5ea'
,'Tamara', 'Sadek', 'Lebanon', current_timestamp),

('JohnSmith', 'JohnSmith@gmail.com',
'd19edeb7198f1e9ee2ef8d1876f49b3ec3c4c3fc126c6419734397741f048d318', 'John', 'Smith', 'USA',
current_timestamp),

('EmmaJohnson', 'EmmaJohnson@gmail.com',
'f01c86dbff968e105264b2dab1b71f0631a36047aa4b0f66a4b51b69cb68ff12', 'Emma', 'Johnson', 'USA',
current_timestamp),

('MichaelWilson', 'MichaelWilson@outlook.com',
'bfc47af6e83f5a3f98f3c5ec42022100b453bd5d6d17c79d05d7482d309558d7', 'Michael', 'Wilson',
'Canada', current_timestamp),

('SophiaJones', 'SophiaJones@mail.com',
'8e11a7fbf6d88abf9c20758bb8c005855542f6c2edc283f8d8d5d9eb61f4c5e6', 'Sophia', 'Jones', 'Canada',
current_timestamp),

('DavidBrown', 'DavidBrown@mail.com',
'0bf10ed62bf8762c753ab8c1b9d9326b53c7cd0491deea8db3f3a6b675400bda', 'David', 'Brown', 'UK',
current_timestamp),

('OliviaDavis', 'OliviaDavis@yahoo.com',
'e6ccf9d13c1167c3a6b61c0176473cd977640999ef974ea83a75be88918046ec', 'Olivia', 'Davis', 'UK',
current_timestamp),

('JamesMiller', 'JamesMiller@gmail.com',
'2a0a3842a5046e301f3e4d24cfb2d5fb924932f6c3c72318d5f1cfa216e68549', 'James', 'Miller',
'Australia', current_timestamp),

('SophiaHarris', 'SophiaHarris@gmail.com',
'cfdad2c74ab40a22fc08c67183ad02ce431826b26d69c650ff8c877ed489b950', 'Sophia', 'Harris',
'Australia', current_timestamp);

username [PK] text	email text	passwod text	firstName text	lastName text	country text	registrationDate date
AliFarhat	AliFarhat@gmail.com	2d3e1312b4f10a8a7c0ebc7dc5ea8cf2cbca99e7c8bd6a57f188e7e1e5571d7	Ali	Farhat	Lebanon	2022-10-21
ay_charaf	ay_charaf@gmail.com	3a63c03f43710ad8eae4c0d6de315df0d73f0b3b5f5ae8e09d4da598b968c66	Ayman	Charaf	Lebanon	2023-06-08
BinaryExplorer88	binbin77@mail.com	SSS8oj1b	Mia	Sanchez	Lebanon	2019-01-10
BlueCoder42	bluecoder@gmail.com	YSBN93u28y382	Alex	Turner	Japan	2023-04-15
CodeMasterX	masterrr23@gmail.com	suhsYV27	Olivia	Chang	Germany	2022-11-22
cutiepie	cutiepie@gmail.com	e2bf8da1bde8ed5e6ccf3f936da984ae11a2c0403c54f64eb69a1e7b1d3d62c	Lama	Hasbini	America	2023-02-21
CyberPioneer	cyber101@gmail.com	dhYGF27	Ava	Thompson	Canada	2020-08-09
Danosour	Danosour@gmail.com	1f548a2d8992b4d8a6cb424f3c6aeef19e719e5d158c6fb39e9ea2ae23f4570	Dana	Kossaybeyati	Lebanon	2022-08-16
DavidBrown	DavidBrown@mail.com	0bf10ed62bf8762c753ab8c1b9d9326b53c7cd0491deea8db3f3a6b675400b...	David	Brown	UK	2023-11-10
EmmaJohnson	EmmaJohnson@gmail.com	f01c86dbff968e105264b2dab1b71f0631a36047aa4b0f66a4b51b69cb68ff12	Emma	Johnson	USA	2023-11-10
HashTrash	HashTrash@gmail.com	f5297c5e46c2f95e5c700f946e77dd715583cfb41895fcb52e4d85b7eb38127	Hashem	Khodor	Lebanon	2023-02-14
HussKiller	HussKiller@gmail.com	ad8b3b7d8755165555a80f8b0b5f196ea44d6c0a54f491b7a2a66e8dfdece0b	Hussein	Bakri	Lebanon	2022-07-30
ImanDestroyer	ImanDestroyer@gmail.com	ea4109f401e9aabbab4c5fb17c25f31e5c64bfa5e1efcdfcf4c9ec9910b896d	Iman	Ghalayini	Lebanon	2023-03-04
JamesMiller	JamesMiller@gmail.com	2a0a3842a5046e301f3e4d24cfb2d5fb924932f6c3c72318d5f1cfa216e68549	James	Miller	Australia	2023-11-10
JohnSmith	JohnSmith@gmail.com	d19edeb7198f1e9ee2ef8d1876f49b3ec3c4c3fc126c6419734397741f048d3...	John	Smith	USA	2023-11-10
Khaled47	Khaled47@gmail.com	5e884898da28047151d0e56f8dc6292773603d0d6b8b4b9b883883a1541d3...	Khaled	Charaf	Lebanon	2023-01-15
MichaelWilson	MichaelWilson@outlook.com	bfc47af6e83f5a3f98f3c5ec42022100b453bd5d6d17c79d05d7482d309558d7	Michael	Wilson	Canada	2023-11-10
MK	MK@gmail.com	6600bf6331c98f1619619fc8e405d5ec3dd97e614b9d241f14275168f130c916	Mohamad	Kreidieh	Lebanon	2023-11-10
MommyBoy	MommyBoy@gmail.com	fd1fc301decbde0a5ad5c4c27ae398d07bb9ec75c7e2c1b11a76d0068d15cd	Abdelatif	Saleh	Lebanon	2023-01-29
OliviaDavis	OliviaDavis@yahoo.com	e6ccf9d13c1167c3a6b61c0176473cd977640999ef974ea83a75be88918046...	Olivia	Davis	UK	2023-11-10
PixelGeek23	pixy77@mail.com	826HVGT2	Liam	Mitchell	India	2020-05-18
QuantumCoder7	quantumcoder@mail.com	482t40	Ethan	Bennett	Australia	2023-07-03
SophiaHarris	SophiaHarris@gmail.com	cfdad2c74ab40a22fc08c67183ad02ce431826b26d69c650ff8c877ed489b950	Sophia	Harris	Australia	2023-11-10
SophiaJones	SophiaJones@mail.com	8e11a7fbf6d88abf9c20758bb8c005855542f6c2edc283f8d8d5d9eb61f4c5e6	Sophia	Jones	Canada	2023-11-10
SwiftSorcerer	swiftt1@mail.com	7S28B2G	Noah	Patel	South Africa	2022-12-30
TamTam	TamTam@gmail.com	6c3bf86f56a3e67d88f6d6bea7e03dd5ca3c31b07a51fc1ce28571e049dce5ea	Tamara	Sadek	Lebanon	2023-11-10
TechNinja91	techtech@mail.com	sUb292372804	Maya	Rodriguez	Brazil	2022-09-28
TheJoker	TheJoker@gmail.com	efbc56e0d798f9372831ceab35f8c3d812ead0c4e1c1ac21f96c0f8bb4249f7	Omar	Ramadan	Lebanon	2023-02-04
Tourist	Tourist@gmail.com	5e884898da28047151d0e56f8dc6292773603d0d6b8b4b9b883883a1541d3...	Gennady	Korotkevich	Belarus	2022-08-16

11.3.2- Befriends:

```
INSERT INTO public."Befriends"(
        "friend1Username", "friend2Username", "timestamp")
        VALUES ('Khaled47', 'AliFarhat', '2023-01-15'),
    ('Khaled47', 'ay_charaf', '2022-10-21'),
    ('Khaled47', 'HussKiller', '2023-06-08'),
    ('Khaled47', 'ImanDestroyer', '2022-07-30'),
    ('AliFarhat', 'ay_charaf', '2023-03-04'),
    ('AliFarhat', 'HussKiller', '2022-08-16'),
    ('AliFarhat', 'ImanDestroyer', '2023-02-14'),
    ('AliFarhat', 'TheJoker', '2023-02-04'),
```

('ay_charaf', 'HussKiller', '2023-01-29'),

('ay_charaf', 'ImanDestroyer', '2023-02-21'),

('HussKiller', 'ImanDestroyer', '2022-08-16'),

('HussKiller', 'HashTrash', '2023-01-15'),

('ImanDestroyer', 'Danosour', '2022-10-21'),

('ImanDestroyer', 'TheJoker', '2023-06-08'),

('Danosour', 'HashTrash', '2022-07-30'),

('Danosour', 'TheJoker', '2023-03-04'),

('HashTrash', 'TheJoker', '2022-08-16'),

('HashTrash', 'MommyBoy', '2023-02-14'),

('TheJoker', 'MommyBoy', '2023-02-04'),

('TheJoker', 'cutiepie', '2023-01-29');

friend1Username [PK] text	friend2Username [PK] text	timestamp timestamp without time zone
AliFarhat	ay_charaf	2023-03-04 00:00:00
AliFarhat	HussKiller	2022-08-16 00:00:00
AliFarhat	ImanDestroyer	2023-02-14 00:00:00
AliFarhat	TheJoker	2023-02-04 00:00:00
ay_charaf	HussKiller	2023-01-29 00:00:00
ay_charaf	ImanDestroyer	2023-02-21 00:00:00
Danosour	HashTrash	2022-07-30 00:00:00
Danosour	TheJoker	2023-03-04 00:00:00
HashTrash	MommyBoy	2023-02-14 00:00:00
HashTrash	TheJoker	2022-08-16 00:00:00
HussKiller	HashTrash	2023-01-15 00:00:00
HussKiller	ImanDestroyer	2022-08-16 00:00:00
ImanDestroyer	Danosour	2022-10-21 00:00:00
ImanDestroyer	TheJoker	2023-06-08 00:00:00
Khaled47	AliFarhat	2023-01-15 00:00:00
Khaled47	ay_charaf	2022-10-21 00:00:00
Khaled47	HussKiller	2023-06-08 00:00:00
Khaled47	ImanDestroyer	2022-07-30 00:00:00
TheJoker	cutiepie	2023-01-29 00:00:00
TheJoker	MommyBoy	2023-02-04 00:00:00

11.3.3- Contest:

```
INSERT INTO public."Contest"(
        "roundNumber", name, divsion, "startTimestamp", length, description)
        VALUES
-- Contest 1
('HusseinBakri Championship', 1, '2023-01-15 08:00:00', '03:00:00', 'The ultimate coding challenge for elite programmers.'),
-- Contest 2
('AUB Showdown', 2, '2023-02-05 10:30:00', '02:30:00', 'Test your algorithmic skills in this exciting contest.'),
-- Contest 3
('LAU Titans Challenge', 3, '2023-03-20 09:15:00', '03:00:00', 'An opportunity for newcomers to prove their tech prowess.'),
-- Contest 4
('CodeCrafters Duel', 1, '2023-04-10 14:00:00', '04:00:00', 'A battle of wits among the coding elite.'),
-- Contest 5
('AlgorithmShip', 2, '2023-05-02 11:45:00', '02:30:00', 'Marvel at the magic of algorithms in this contest and get a Scholarship.'),
-- Contest 6
('Rookie Coding Challenge', 3, '2023-06-15 10:00:00', '03:00:00', 'An ideal platform for budding programmers to shine.'),
-- Contest 7
('CodeFusion Challenge', 1, '2023-07-08 13:30:00', '04:00:00', 'Merge your coding skills in this fusion of challenges.'),
-- Contest 8
```

('Algorithm Explorers', 2, '2023-08-19 11:15:00', '02:30:00', 'Explore the depths of algorithms in this exciting contest.'),

-- Contest 9

('Dakowdas Enthusiasts', 3, '2023-09-25 09:45:00', '03:00:00', 'A showdown for tech enthusiasts to conquer.'),

-- Contest 10

('Bakri Finale', 1, '2023-10-15 14:15:00', '04:00:00', 'Place top 1 to get a 100 on Bakris database course at AUB!');

roundNumber [PK] integer	name text	divsion integer	startTimestamp timestamp without time zone	length time without time zone	description text
1	HusseinBakri Championship	1	2023-01-15 08:00:00	03:00:00	The ultimate coding challenge for elite programmers.
2	AUB Showdown	2	2023-02-05 10:30:00	02:30:00	Test your algorithmic skills in this exciting contest.
3	LAU Titans Challenge	3	2023-03-20 09:15:00	03:00:00	An opportunity for newcomers to prove their tech prowess.
4	CodeCrafters Duel	1	2023-04-10 14:00:00	04:00:00	A battle of wits among the coding elite.
5	AlgorithmShip	2	2023-05-02 11:45:00	02:30:00	Marvel at the magic of algorithms in this contest and get a Scholarsh...
6	Rookie Coding Challenge	3	2023-06-15 10:00:00	03:00:00	An ideal platform for budding programmers to shine.
7	CodeFusion Challenge	1	2023-07-08 13:30:00	04:00:00	Merge your coding skills in this fusion of challenges.
8	Algorithm Explorers	2	2023-08-19 11:15:00	02:30:00	Explore the depths of algorithms in this exciting contest.
9	Dakowdas Enthusiasts	3	2023-09-25 09:45:00	03:00:00	A showdown for tech enthusiasts to conquer.
10	Bakri Finale	1	2023-10-15 14:15:00	04:00:00	Place top 1 to get a 100 on Bakris database course at AUB!

11.3.4- ProgrammingProblem:

INSERT INTO public."ProgrammingProblem"("problemID", "difficultyLevel", title, description, "timeLimit", "memoryLimit", "roundNumber")

VALUES

-- Round 1

('A', 'Two Sum Problem', 'Given an array of integers and a target sum, find two numbers that add up to the target sum. Return the indices of the two elements.', 2, 256, 1),

('B', 'Palindrome Check', 'Write a program to check if a given string is a palindrome or not. Consider alphanumeric characters and ignore case.', 3, 512, 1),

('C', 'Maximum Subarray', 'Find the contiguous subarray with the largest sum in an array of integers. Return the starting and ending indices of the subarray.', 4, 128, 1),

('D', 'Factorial Calculation', 'Write a program to calculate the factorial of a given non-negative integer. Handle large factorials efficiently.', 5, 256, 1),

('E', 'Graph Traversal', 'Traverse a graph and perform depth-first or breadth-first search. Implement both DFS and BFS algorithms for a given graph.', 1, 512, 1),

('F', '0/1 Knapsack Problem', 'Solve the 0/1 Knapsack problem using dynamic programming. Given a set of items with weights and values, maximize the value within a limited weight capacity.', 4, 256, 1),

-- Sortings , BruteForce, Greedy , DP , Graphs , DP

-- Round 2

('A', 'Greatest Common Divisor', 'Find the greatest common divisor (GCD) of two positive integers using an efficient algorithm.', 1, 128, 2),

('B', 'Binary Tree Height', 'Calculate the height of a binary tree, which is the length of the longest path from the root to a leaf node.', 2, 512, 2),

('C', 'Longest Common Substring', 'Find the longest common substring between two given strings. Return the substring itself.', 3, 256, 2),

('D', 'Sparse Matrix Multiplication', 'Perform multiplication of two sparse matrices efficiently, taking advantage of their sparsity.', 4, 128, 2),

('E', 'Shortest Path in Maze', 'Find the shortest path from the start to the end in a maze using breadth-first search.', 5, 512, 2),

('F', 'Matrix Chain Multiplication', 'Solve the matrix chain multiplication problem using dynamic programming to minimize multiplication cost.', 1, 256, 2),

-- NumberTheory , Graphs , DP ,Matrices,Graphs,DiveAndConquer

-- Round 3

('A', 'Rotated Sorted Array Search', 'Search for a target element in a rotated sorted array efficiently using a modified binary search.', 2, 256, 3),

('B', 'Binary Tree Traversal', 'Perform various traversals (inorder, preorder, postorder) on a binary tree and print the node values.', 3, 512, 3),

('C', 'Regular Expression Matcher', 'Implement a regular expression matcher that can match strings against regular expressions.', 4, 128, 3),

('D', 'Sudoku Solver', 'Solve a Sudoku puzzle by filling in the empty cells with the correct numbers while adhering to the rules.', 5, 256, 3),

('E', 'Chess Game Logic', 'Implement the logic for a basic chess game, including moving pieces and checking for checkmate conditions.', 1, 512, 3),

('F', 'Longest Common Subsequence', 'Find the longest common subsequence of two given strings using dynamic programming.', 4, 256, 3),

-- Binary Search , Graphs, Graphs , Brute Force, Graphs, DP

-- Round 4

('A', 'Longest Increasing Subsequence', 'Find the length of the longest increasing subsequence in an array of integers using dynamic programming.', 1, 128, 4),

('B', 'Shortest Path', 'Implement Dijkstras algorithm to find the shortest path between two nodes in a weighted graph.', 2, 512, 4),

('C', 'Knuth-Morris-Pratt', 'Write an algorithm to search for a pattern in a text using the Knuth-Morris-Pratt string matching algorithm.', 3, 256, 4),

('D', 'Prime Factorization', 'Factorize a given integer into its prime factors and their exponents.', 4, 128, 4),

('E', 'Maximum Subarray Sum', 'Find the maximum sum of a contiguous subarray within an array of integers using dynamic programming.', 5, 512, 4),

('F', 'Traveling Salesman', 'Solve the Traveling Salesman Problem using dynamic programming to find the shortest tour that visits a set of cities.', 1, 256, 4),

-- DP , Graphs , String Matching , NumberTheory , DP , DP

-- Round 5

('A', 'Longest Palindromic Substring', 'Find the longest palindromic substring in a given string using dynamic programming.', 1, 128, 5),

('B', 'Minimum Spanning Tree', 'Implement Kruskals algorithm to find the minimum spanning tree of a connected graph.', 2, 512, 5),

('C', 'Boyer-Moore', 'Write an algorithm to search for a pattern in a text using the Boyer-Moore string matching algorithm.', 3, 256, 5),

('D', 'Modular Exponentiation', 'Compute large powers modulo a prime number efficiently using modular exponentiation.', 4, 128, 5),

('E', 'Coin Change', 'Find the number of ways to make change for a given amount using a set of coin denominations.', 5, 512, 5),

('F', 'Longest Common Subarray', 'Find the longest common subarray between two arrays using dynamic programming.', 1, 256, 5),

-- DP , Graphs , String Matching , Number Theory , DP , DP

-- Round 6

('A', 'Maximum Subarray Sum (Non-Contiguous)', 'Find the maximum sum of a non-contiguous subarray within an array of integers using dynamic programming.', 2, 256, 6),

('B', 'Topological Sorting', 'Implement topological sorting for directed acyclic graphs to find a linear order of vertices.', 3, 512, 6),

('C', 'Rabin-Karp', 'Write an algorithm to search for a pattern in a text using the Rabin-Karp string matching algorithm.', 4, 256, 6),

('D', 'Greatest Common Divisor', 'Find the greatest common divisor (GCD) of two integers using the Euclidean algorithm.', 5, 128, 6),

('E', 'Longest Increasing Subsequence', 'Find the length of the longest increasing subsequence in an array of integers using dynamic programming.', 1, 512, 6),

('F', 'Bipartite Graph Check', 'Determine if a given graph is bipartite or not using graph coloring algorithms.', 2, 256, 6),

-- DP , Graphs , String Matching , NumberTheory, DP, Graphs

-- Round 7

('A', 'Matrix Chain Multiplication', 'Solve the matrix chain multiplication problem using dynamic programming to minimize multiplication cost.', 3, 128, 7),

('B', 'Strongly Connected Components', 'Implement Tarjans algorithm to find strongly connected components in a directed graph.', 4, 512, 7),

('C', 'Aho-Corasick', 'Write an algorithm to search for multiple patterns in a text using the Aho-Corasick string matching algorithm.', 5, 256, 7),

('D', 'Sieve of Eratosthenes', 'Generate prime numbers up to a given limit using the Sieve of Eratosthenes algorithm.', 1, 128, 7),

('E', 'Longest Common Subsequence', 'Find the longest common subsequence of two sequences using dynamic programming.', 2, 512, 7),

('F', 'Shortest Path (Negative Weight)', 'Implement the Bellman-Ford algorithm to find the shortest path in a graph with negative weight edges.', 3, 256, 7),

-- DP , Graphs, String Matching , NumberTheory , DP , Graphs

-- Round 8

('A', 'Coin Change (Minimum Coins)', 'Find the minimum number of coins needed to make change for a given amount using dynamic programming.', 4, 128, 8),

('B', 'Articulation Points and Bridges', 'Find articulation points and bridges in an undirected graph using depth-first search.', 5, 512, 8),

('C', 'Longest Common Prefix Array', 'Compute the longest common prefix array for a set of strings to optimize string matching.', 1, 256, 8),

('D', 'Modular Multiplicative Inverse', 'Calculate the modular multiplicative inverse of an integer modulo a prime using the extended Euclidean algorithm.', 2, 128, 8),

('E', 'Rod Cutting', 'Solve the rod cutting problem to maximize profit using dynamic programming.', 3, 512, 8),

('F', 'Maximum Flow', 'Find the maximum flow in a flow network using the Ford-Fulkerson algorithm.', 4, 256, 8),

-- DP , Graphs , String Matching, NumberTheory, DP , Graphs

-- Round 9

('A', 'Longest Palindromic Subsequence', 'Find the length of the longest palindromic subsequence in a given string using dynamic programming.', 5, 128, 9),

('B', 'Minimum Spanning Tree (Prims Algorithm)', 'Implement Prims algorithm to find the minimum spanning tree of a connected graph.', 1, 512, 9),

('C', 'Z Algorithm', 'Compute the Z array to efficiently search for a pattern in a text using the Z algorithm.', 2, 256, 9),

('D', 'Chinese Remainder Theorem', 'Solve simultaneous modular congruences using the Chinese Remainder Theorem.', 3, 128, 9),

('E', 'Edit Distance', 'Find the edit distance (Levenshtein distance) between two strings using dynamic programming.', 4, 512, 9),

('F', 'Hamiltonian Path', 'Find a Hamiltonian path in a directed or undirected graph using backtracking or dynamic programming.', 5, 256, 9),

-- DP, Graphs, String Matching, NumberTheory, DP, Graphs

-- Round 10

('A', 'Longest Zigzag Subsequence', 'Find the length of the longest zigzag subsequence in an array of integers using dynamic programming.', 1, 128, 10),

('B', 'Maximum Bipartite Matching', 'Find the maximum cardinality matching in a bipartite graph using augmenting paths.', 2, 512, 10),

('C', 'Suffix Array', 'Construct a suffix array for a given string to enable efficient substring searches and pattern matching.', 3, 256, 10),

('D', 'Lucas Theorem', 'Apply Lucass Theorem to compute binomial coefficients modulo a prime number.', 4, 128, 10),

('E', 'Longest Increasing Subarray', 'Find the longest increasing subarray within an array of integers using dynamic programming.', 5, 512, 10),

('F', 'Maximum Planar Subgraph', 'Find the maximum planar subgraph of a planar graph using planarity testing and Kuratowskis theorem.', 1, 256, 10);

-- DP , Graph , String Matching , NumberTheory , DP, Graph

problemID [PK] integer	difficultyLevel character	title text	description text	timeLimit integer	memoryLimit integer	roundNumber integer
1	A	Two Sum Problem	Given an array of integers and a target sum, find two numbers that add up to the target sum. Return th...	2	256	1
2	B	Palindrome Check	Write a program to check if a given string is a palindrome or not. Consider alphanumeric characters an...	3	512	1
3	C	Maximum Subarray	Find the contiguous subarray with the largest sum in an array of integers. Return the starting and endin...	4	128	1
4	D	Factorial Calculation	Write a program to calculate the factorial of a given non-negative integer. Handle large factorials effici...	5	256	1
5	E	Graph Traversal	Traverse a graph and perform depth-first or breadth-first search. Implement both DFS and BFS algorith...	1	512	1
6	F	0/1 Knapsack Problem	Solve the 0/1 Knapsack problem using dynamic programming. Given a set of items with weights and v...	4	256	1
7	A	Greatest Common Divisor	Find the greatest common divisor (GCD) of two positive integers using an efficient algorithm.	1	128	2
8	B	Binary Tree Height	Calculate the height of a binary tree, which is the length of the longest path from the root to a leaf node.	2	512	2
9	C	Longest Common Substring	Find the longest common substring between two given strings. Return the substring itself.	3	256	2
10	D	Sparse Matrix Multiplication	Perform multiplication of two sparse matrices efficiently, taking advantage of their sparsity.	4	128	2
11	E	Shortest Path in Maze	Find the shortest path from the start to the end in a maze using breadth-first search.	5	512	2
12	F	Matrix Chain Multiplication	Solve the matrix chain multiplication problem using dynamic programming to minimize multiplication c...	1	256	2
13	A	Rotated Sorted Array Search	Search for a target element in a rotated sorted array efficiently using a modified binary search.	2	256	3
14	B	Binary Tree Traversal	Perform various traversals (inorder, preorder, postorder) on a binary tree and print the node values.	3	512	3
15	C	Regular Expression Matcher	Implement a regular expression matcher that can match strings against regular expressions.	4	128	3
16	D	Sudoku Solver	Solve a Sudoku puzzle by filling in the empty cells with the correct numbers while adhering to the rules.	5	256	3
17	E	Chess Game Logic	Implement the logic for a basic chess game, including moving pieces and checking for checkmate con...	1	512	3
18	F	Longest Common Subsequence	Find the longest common subsequence of two given strings using dynamic programming.	4	256	3
19	A	Longest Increasing Subsequence	Find the length of the longest increasing subsequence in an array of integers using dynamic programm...	1	128	4
20	B	Shortest Path	Implement Dijkstras algorithm to find the shortest path between two nodes in a weighted graph.	2	512	4
21	C	Knuth-Morris-Pratt	Write an algorithm to search for a pattern in a text using the Knuth-Morris-Pratt string matching algorit...	3	256	4
22	D	Prime Factorization	Factorize a given integer into its prime factors and their exponents.	4	128	4
23	E	Maximum Subarray Sum	Find the maximum sum of a contiguous subarray within an array of integers using dynamic programmi...	5	512	4
24	F	Traveling Salesman	Solve the Traveling Salesman Problem using dynamic programming to find the shortest tour that visits ...	1	256	4
25	A	Longest Palindromic Substring	Find the longest palindromic substring in a given string using dynamic programming.	1	128	5
26	B	Minimum Spanning Tree	Implement Kruskals algorithm to find the minimum spanning tree of a connected graph.	2	512	5
27	C	Boyer-Moore	Write an algorithm to search for a pattern in a text using the Boyer-Moore string matching algorithm.	3	256	5
28	D	Modular Exponentiation	Compute large powers modulo a prime number efficiently using modular exponentiation.	4	128	5
29	E	Coin Change	Find the number of ways to make change for a given amount using a set of coin denominations.	5	512	5
30	F	Longest Common Subarray	Find the longest common subarray between two arrays using dynamic programming.	1	256	5
31	A	Maximum Subarray Sum (Non-Contiguous)	Find the maximum sum of a non-contiguous subarray within an array of integers using dynamic progra...	2	256	6
32	B	Topological Sorting	Implement topological sorting for directed acyclic graphs to find a linear order of vertices.	3	512	6
33	C	Rabin-Karp	Write an algorithm to search for a pattern in a text using the Rabin-Karp string matching algorithm.	4	256	6
34	D	Greatest Common Divisor	Find the greatest common divisor (GCD) of two integers using the Euclidean algorithm.	5	128	6
35	E	Longest Increasing Subsequence	Find the length of the longest increasing subsequence in an array of integers using dynamic programm...	1	512	6
36	F	Bipartite Graph Check	Determine if a given graph is bipartite or not using graph coloring algorithms.	2	256	6
37	A	Matrix Chain Multiplication	Solve the matrix chain multiplication problem using dynamic programming to minimize multiplication c...	3	128	7
38	B	Strongly Connected Components	Implement Tarjans algorithm to find strongly connected components in a directed graph.	4	512	7
39	C	Aho-Corasick	Write an algorithm to search for multiple patterns in a text using the Aho-Corasick string matching alg...	5	256	7
40	D	Sieve of Eratosthenes	Generate prime numbers up to a given limit using the Sieve of Eratosthenes algorithm.	1	128	7
41	E	Longest Common Subsequence	Find the longest common subsequence of two sequences using dynamic programming.	2	512	7
42	F	Shortest Path (Negative Weight)	Implement the Bellman-Ford algorithm to find the shortest path in a graph with negative weight edges.	3	256	7
43	A	Coin Change (Minimum Coins)	Find the minimum number of coins needed to make change for a given amount using dynamic progra...	4	128	8
44	B	Articulation Points and Bridges	Find articulation points and bridges in an undirected graph using depth-first search.	5	512	8
45	C	Longest Common Prefix Array	Compute the longest common prefix array for a set of strings to optimize string matching.	1	256	8
46	D	Modular Multiplicative Inverse	Calculate the modular multiplicative inverse of an integer modulo a prime using the extended Euclidea...	2	128	8
47	E	Rod Cutting	Solve the rod cutting problem to maximize profit using dynamic programming.	3	512	8
48	F	Maximum Flow	Find the maximum flow in a flow network using the Ford-Fulkerson algorithm.	4	256	8
49	A	Longest Palindromic Subsequence	Find the length of the longest palindromic subsequence in a given string using dynamic programming.	5	128	9
50	B	Minimum Spanning Tree (Prims Algorithm)	Implement Prims algorithm to find the minimum spanning tree of a connected graph.	1	512	9
51	C	Z Algorithm	Compute the Z array to efficiently search for a pattern in a text using the Z algorithm.	2	256	9
52	D	Chinese Remainder Theorem	Solve simultaneous modular congruences using the Chinese Remainder Theorem.	3	128	9
53	E	Edit Distance	Find the edit distance (Levenshtein distance) between two strings using dynamic programming.	4	512	9
54	F	Hamiltonian Path	Find a Hamiltonian path in a directed or undirected graph using backtracking or dynamic programming.	5	256	9
55	A	Longest Zigzag Subsequence	Find the length of the longest zigzag subsequence in an array of integers using dynamic programming.	1	128	10
56	B	Maximum Bipartite Matching	Find the maximum cardinality matching in a bipartite graph using augmenting paths.	2	512	10
57	C	Suffix Array	Construct a suffix array for a given string to enable efficient substring searches and pattern matching.	3	256	10
58	D	Lucas Theorem	Apply Lucass Theorem to compute binomial coefficients modulo a prime number.	4	128	10
59	E	Longest Increasing Subarray	Find the longest increasing subarray within an array of integers using dynamic programming.	5	512	10
60	F	Maximum Planar Subgraph	Find the maximum planar subgraph of a planar graph using planarity testing and Kuratowskis theorem.	1	256	10

11.3.5- Solutions:

```
INSERT INTO public."Solution"("problemID", "sourceCode")

VALUES

(1, '#include <iostream> using namespace std ; int main(){//solution for Problem 1  return 0 ;}'),

(2, '#include <iostream> using namespace std ; int main(){//solution for Problem 2  return 0 ;}'),

(3, '#include <iostream> using namespace std ; int main(){//solution for Problem 3  return 0 ;}'),

(4, '#include <iostream> using namespace std ; int main(){//solution for Problem 4  return 0 ;}'),

(5, '#include <iostream> using namespace std ; int main(){//solution for Problem 5  return 0 ;}'),

(6, '#include <iostream> using namespace std ; int main(){//solution for Problem 6  return 0 ;}'),

(7, '#include <iostream> using namespace std ; int main(){//solution for Problem 7  return 0 ;}'),

(8, '#include <iostream> using namespace std ; int main(){//solution for Problem 8  return 0 ;}'),

(9, '#include <iostream> using namespace std ; int main(){//solution for Problem 9  return 0 ;}'),

(10, '#include <iostream> using namespace std ; int main(){//solution for Problem 10  return 0 ;}'),

(11, '#include <iostream> using namespace std ; int main(){//solution for Problem 11  return 0 ;}'),

(12, '#include <iostream> using namespace std ; int main(){//solution for Problem 12  return 0 ;}'),

(13, '#include <iostream> using namespace std ; int main(){//solution for Problem 13  return 0 ;}'),

(14, '#include <iostream> using namespace std ; int main(){//solution for Problem 14  return 0 ;}'),

(15, '#include <iostream> using namespace std ; int main(){//solution for Problem 15  return 0 ;}'),

(16, '#include <iostream> using namespace std ; int main(){//solution for Problem 16  return 0 ;}'),

(17,'#include <iostream> using namespace std ; int main(){//solution for Problem 17  return 0 ;}'),

(18, '#include <iostream> using namespace std ; int main(){//solution for Problem 18  return 0 ;}'),

(19, '#include <iostream> using namespace std ; int main(){//solution for Problem 25  return 0 ;}'),

(20, '#include <iostream> using namespace std ; int main(){//solution for Problem 26  return 0 ;}'),
```

(21, '#include <iostream> using namespace std ; int main(){//solution for Problem 27 return 0 ;}'),

(22, '#include <iostream> using namespace std ; int main(){//solution for Problem 28 return 0 ;}'),

(23, '#include <iostream> using namespace std ; int main(){//solution for Problem 29 return 0 ;}'),

(24, '#include <iostream> using namespace std ; int main(){//solution for Problem 30 return 0 ;}'),

(25, '#include <iostream> using namespace std ; int main(){//solution for Problem 31 return 0 ;}'),

(26, '#include <iostream> using namespace std ; int main(){//solution for Problem 32 return 0 ;}'),

(27, '#include <iostream> using namespace std ; int main(){//solution for Problem 33 return 0 ;}'),

(28, '#include <iostream> using namespace std ; int main(){//solution for Problem 34 return 0 ;}'),

(29, '#include <iostream> using namespace std ; int main(){//solution for Problem 35 return 0 ;}'),

(30, '#include <iostream> using namespace std ; int main(){//solution for Problem 36 return 0 ;}'),

(31, '#include <iostream> using namespace std ; int main(){//solution for Problem 37 return 0 ;}'),

(32, '#include <iostream> using namespace std ; int main(){//solution for Problem 38 return 0 ;}'),

(33, '#include <iostream> using namespace std ; int main(){//solution for Problem 39 return 0 ;}'),

(34, '#include <iostream> using namespace std ; int main(){//solution for Problem 40 return 0 ;}'),

(35, '#include <iostream> using namespace std ; int main(){//solution for Problem 41 return 0 ;}'),

(36,'#include <iostream> using namespace std ; int main(){//solution for Problem 42 return 0 ;}'),

(37,'#include <iostream> using namespace std ; int main(){//solution for Problem 43 return 0 ;}')

(38, '#include <iostream> using namespace std ; int main(){//solution for Problem 44 return 0 ;}'),

(39, '#include <iostream> using namespace std ; int main(){//solution for Problem 45 return 0 ;}'),

(40, '#include <iostream> using namespace std ; int main(){//solution for Problem 46 return 0 ;}'),

(41, '#include <iostream> using namespace std ; int main(){//solution for Problem 47 return 0 ;}'),

(42, '#include <iostream> using namespace std ; int main(){//solution for Problem 48 return 0 ;}'),

(43, '#include <iostream> using namespace std ; int main(){//solution for Problem 49 return 0 ;}'),

(44, '#include <iostream> using namespace std ; int main(){//solution for Problem 50 return 0 ;}'),

(45, '#include <iostream> using namespace std ; int main(){//solution for Problem 51 return 0 ;}'),

(46, '#include <iostream> using namespace std ; int main(){//solution for Problem 52 return 0 ;}'),

(47, '#include <iostream> using namespace std ; int main(){//solution for Problem 53 return 0 ;}'),

(48, '#include <iostream> using namespace std ; int main(){//solution for Problem 54 return 0 ;}'),

(49,'#include <iostream> using namespace std ; int main(){//solution for Problem 55 return 0 ;}'),

(50, '#include <iostream> using namespace std ; int main(){//solution for Problem 56 return 0 ;}'),

(51, '#include <iostream> using namespace std ; int main(){//solution for Problem 57 return 0 ;}'),

(52, '#include <iostream> using namespace std ; int main(){//solution for Problem 58 return 0 ;}'),

(53, '#include <iostream> using namespace std ; int main(){//solution for Problem 59 return 0 ;}'),

(54, '#include <iostream> using namespace std ; int main(){//solution for Problem 60 return 0 ;}'),

(55, '#include <iostream> using namespace std ; int main(){//solution for Problem 61 return 0 ;}'),

(56, '#include <iostream> using namespace std ; int main(){//solution for Problem 62 return 0 ;}'),

(57, '#include <iostream> using namespace std ; int main(){//solution for Problem 63 return 0 ;}'),

(58, '#include <iostream> using namespace std ; int main(){//solution for Problem 64 return 0 ;}'),

(59, '#include <iostream> using namespace std ; int main(){//solution for Problem 65 return 0 ;}'),

(60,'#include <iostream> using namespace std ; int main(){//solution for Problem 66 return 0 ;}');

problemID [PK] integer	solutionID [PK] integer	sourceCode text
1	1	#include <iostream> using namespace std ; int main(){//solution for Problem 1 return 0 ;}
2	2	#include <iostream> using namespace std ; int main(){//solution for Problem 2 return 0 ;}
3	3	#include <iostream> using namespace std ; int main(){//solution for Problem 3 return 0 ;}
4	4	#include <iostream> using namespace std ; int main(){//solution for Problem 4 return 0 ;}
5	5	#include <iostream> using namespace std ; int main(){//solution for Problem 5 return 0 ;}
6	6	#include <iostream> using namespace std ; int main(){//solution for Problem 6 return 0 ;}
7	7	#include <iostream> using namespace std ; int main(){//solution for Problem 7 return 0 ;}
8	8	#include <iostream> using namespace std ; int main(){//solution for Problem 8 return 0 ;}
9	9	#include <iostream> using namespace std ; int main(){//solution for Problem 9 return 0 ;}
10	10	#include <iostream> using namespace std ; int main(){//solution for Problem 10 return ...
11	11	#include <iostream> using namespace std ; int main(){//solution for Problem 11 return ...
12	12	#include <iostream> using namespace std ; int main(){//solution for Problem 12 return ...
13	13	#include <iostream> using namespace std ; int main(){//solution for Problem 13 return ...
14	14	#include <iostream> using namespace std ; int main(){//solution for Problem 14 return ...
15	15	#include <iostream> using namespace std ; int main(){//solution for Problem 15 return ...
16	16	#include <iostream> using namespace std ; int main(){//solution for Problem 16 return ...
17	17	#include <iostream> using namespace std ; int main(){//solution for Problem 17 return ...
18	18	#include <iostream> using namespace std ; int main(){//solution for Problem 18 return ...
19	19	#include <iostream> using namespace std ; int main(){//solution for Problem 25 return ...
20	20	#include <iostream> using namespace std ; int main(){//solution for Problem 26 return ...
21	21	#include <iostream> using namespace std ; int main(){//solution for Problem 27 return ...
22	22	#include <iostream> using namespace std ; int main(){//solution for Problem 28 return ...
23	23	#include <iostream> using namespace std ; int main(){//solution for Problem 29 return ...
24	24	#include <iostream> using namespace std ; int main(){//solution for Problem 30 return ...
25	25	#include <iostream> using namespace std ; int main(){//solution for Problem 31 return ...
26	26	#include <iostream> using namespace std ; int main(){//solution for Problem 32 return ...
27	27	#include <iostream> using namespace std ; int main(){//solution for Problem 33 return ...
28	28	#include <iostream> using namespace std ; int main(){//solution for Problem 34 return ...
29	29	#include <iostream> using namespace std ; int main(){//solution for Problem 35 return ...
30	30	#include <iostream> using namespace std ; int main(){//solution for Problem 36 return ...
31	31	#include <iostream> using namespace std ; int main(){//solution for Problem 37 return ...
32	32	#include <iostream> using namespace std ; int main(){//solution for Problem 38 return ...
33	33	#include <iostream> using namespace std ; int main(){//solution for Problem 39 return ...
34	34	#include <iostream> using namespace std ; int main(){//solution for Problem 40 return ...
35	35	#include <iostream> using namespace std ; int main(){//solution for Problem 41 return ...
36	36	#include <iostream> using namespace std ; int main(){//solution for Problem 42 return ...
37	37	#include <iostream> using namespace std ; int main(){//solution for Problem 43 return ...
38	38	#include <iostream> using namespace std ; int main(){//solution for Problem 44 return ...
39	39	#include <iostream> using namespace std ; int main(){//solution for Problem 45 return ...
40	40	#include <iostream> using namespace std ; int main(){//solution for Problem 46 return ...
41	41	#include <iostream> using namespace std ; int main(){//solution for Problem 47 return ...
42	42	#include <iostream> using namespace std ; int main(){//solution for Problem 48 return ...

43	43	#include <iostream> using namespace std ; int main(){//solution for Problem 49 return ...
44	44	#include <iostream> using namespace std ; int main(){//solution for Problem 50 return ...
45	45	#include <iostream> using namespace std ; int main(){//solution for Problem 51 return ...
46	46	#include <iostream> using namespace std ; int main(){//solution for Problem 52 return ...
47	47	#include <iostream> using namespace std ; int main(){//solution for Problem 53 return ...
48	48	#include <iostream> using namespace std ; int main(){//solution for Problem 54 return ...
49	49	#include <iostream> using namespace std ; int main(){//solution for Problem 55 return ...
50	50	#include <iostream> using namespace std ; int main(){//solution for Problem 56 return ...
51	51	#include <iostream> using namespace std ; int main(){//solution for Problem 57 return ...
52	52	#include <iostream> using namespace std ; int main(){//solution for Problem 58 return ...
53	53	#include <iostream> using namespace std ; int main(){//solution for Problem 59 return ...
54	54	#include <iostream> using namespace std ; int main(){//solution for Problem 60 return ...
55	55	#include <iostream> using namespace std ; int main(){//solution for Problem 61 return ...
56	56	#include <iostream> using namespace std ; int main(){//solution for Problem 62 return ...
57	57	#include <iostream> using namespace std ; int main(){//solution for Problem 63 return ...
58	58	#include <iostream> using namespace std ; int main(){//solution for Problem 64 return ...
59	59	#include <iostream> using namespace std ; int main(){//solution for Problem 65 return ...
60	60	#include <iostream> using namespace std ; int main(){//solution for Problem 66 return ...

11.3.6- TestCase:

```
INSERT INTO public."TestCase"("problemID","input", "expectedOutput")

VALUES

(1, '2 7', '9'),

(2, 'racecar', '1'),

(3, '-2 1 3 -4 5', '3'),

(4, '6', '720'),

(5, '0 1 2 3 4', '0 1 3 2 4'),

(6, '4 7', '11'),

(7, '10 20 30', '10'),

(8, '5', '1'),
```

(9, 'programming contest', '0'),

(10, '1 2 3 4 5', '15'),

(11, '8 5', '40'),

(12, '10 20 30', '10 20 30'),

(13, 'programming', 'false'),

(14, 'algorithm is fun', 'true'),

(15, '4 7 3 2 8', '9 2 6 4 10'),

(16, '5 2 1 8 6 3', '5 4 5 9 10 11'),

(17, '15 10 5', '5'),

(18, '4 5 6 7', '4 3 2 1'),

(19, '3', '4'),

(20, '4 6 7', '9'),

(21, 'programming', 'false'),

(22, 'algorithm is fun', 'true'),

(23, '5 7 9', '5'),

(24, '10 20 30', '10'),

(25, 'racecar', '1'),

(26, 'hello', '0'),

(27, 'level', '1'),

(28, '10 20 30', '30'),

(29, '-2 1 3 -4 5', '3'),

(30, '5', '15'),

(31, '0 1 2 3 4', '5'),

(32, '4 7', '8'),

(33, '10 20 30', '20'),

(34, '6', '15'),

(35, '5 10', '6'),

(36, '8 16', '6'),

(37, '7 11', '1'),

(38, '12 18', '9'),

(39, '6', '20'),

(40, '0', '1'),

(41, '20', '683'),

(42, 'programming', 'false'),

(43, 'coding is fun', 'true'),

(44, '5 4 3 2 1', '5 4 3 2 1'),

(45, '1 2 3 4 5', '120'),

(46, '0 0 0 0 0', '1'),

(47, '8 5', '40'),

(48, '10 20 30', '30'),

(49, 'programming contest', '0'),

(50, 'racecar', '1'),

(51, 'hello', '0'),

(52, 'level', '1'),

(53, '10 20 30', '30'),

(54, 'algorithm is fun', '1'),

(55, '12 18', '36'),

(56, '7 21', '7'),

(57, '9 12', '3'),

(58, '5 2 1 8 6 3', '11'),

(59, '15 10 5', '10'),

(60, '4 5 6 7', '4');

problemID [PK] integer	testCaseNumber [PK] integer	input text	expectedOutput text
1	1	2 7	9
2	2	racecar	1
3	3	-2 1 3 -4 5	3
4	4	6	720
5	5	0 1 2 3 4	0 1 3 2 4
6	6	4 7	11
7	7	10 20 30	10
8	8	5	1
9	9	programming contest	0
10	10	1 2 3 4 5	15
11	11	8 5	40
12	12	10 20 30	10 20 30
13	13	programming	false
14	14	algorithm is fun	true
15	15	4 7 3 2 8	9 2 6 4 10
16	16	5 2 1 8 6 3	5 4 5 9 10 11
17	17	15 10 5	5
18	18	4 5 6 7	4 3 2 1
19	19	3	4
20	20	4 6 7	9
21	21	programming	false
22	22	algorithm is fun	true
23	23	5 7 9	5
24	24	10 20 30	10
25	25	racecar	1
26	26	hello	0
27	27	level	1
28	28	10 20 30	30
29	29	-2 1 3 -4 5	3
30	30	5	15
31	31	0 1 2 3 4	5
32	32	4 7	8
33	33	10 20 30	20
34	34	6	15
35	35	5 10	6

36	36	8 16	6
37	37	7 11	1
38	38	12 18	9
39	39	6	20
40	40	0	1
41	41	20	683
42	42	programming	false
43	43	coding is fun	true
44	44	5 4 3 2 1	5 4 3 2 1
45	45	1 2 3 4 5	120
46	46	0 0 0 0 0	1
47	47	8 5	40
48	48	10 20 30	30
49	49	programming contest	0
50	50	racecar	1
51	51	hello	0
52	52	level	1
53	53	10 20 30	30
54	54	algorithm is fun	1
55	55	12 18	36
56	56	7 21	7
57	57	9 12	3
58	58	5 2 1 8 6 3	11
59	59	15 10 5	10
60	60	4 5 6 7	4

11.3.7- Tag:

```
INSERT INTO public."Tag"("problemID", "tag")

VALUES

-- Tags for Problems 1 to 18 and 25 to 66

(1, 'Sortings'),
```

(2, 'BruteForce'),

(3, 'Greedy'),

(4, 'DP'),

(5, 'Graphs'),

(6, 'DP'),

(7, 'NumberTheory'),

(8, 'Graphs'),

(9, 'DP'),

(10, 'Matrices'),

(11, 'Graphs'),

(12, 'DivideAndConquer'),

(13, 'BinarySearch'),

(14, 'Graphs'),

(15, 'Graphs'),

(16, 'BruteForce'),

(17, 'Graphs'),

(18, 'DP'),

(19, 'DP'),

(20, 'NumberTheory'),

(21, 'DP'),

(22, 'DP'),

(23, 'Graphs'),

(24, 'StringMatching'),

(25, 'NumberTheory'),

(26, 'DP'),

(27, 'DP'),

(28, 'DP'),

(29, 'Graphs'),

(30, 'StringMatching'),

(31, 'NumberTheory'),

(32, 'DP'),

(33, 'Graphs'),

(34, 'DP'),

(35, 'Graphs'),

(36, 'StringMatching'),

(37, 'NumberTheory'),

(38, 'DP'),

(39, 'Graphs'),

(40, 'DP'),

(41, 'Graphs'),

(42, 'StringMatching'),

(43, 'NumberTheory'),

(44, 'DP'),

(45, 'Graphs'),

(46, 'DP'),

(47, 'Graphs'),

(48, 'StringMatching'),

(49, 'NumberTheory'),

(50, 'DP'),

(51, 'Graphs'),

(52, 'DP'),

(53, 'Graphs'),

(54, 'StringMatching'),

(55, 'NumberTheory'),

(56, 'DP'),

(57, 'Graphs'),

(58, 'DP'),

(59, 'Graphs'),

(60, 'StringMatching');

problemID [PK] integer	tag [PK] text
1	Sortings
2	BruteForce
3	Greedy
4	DP
5	Graphs
6	DP
7	NumberTheory
8	Graphs
9	DP
10	Matrices
11	Graphs
12	DivideAndConquer
13	BinarySearch
14	Graphs
15	Graphs
16	BruteForce
17	Graphs
18	DP
19	DP
20	NumberTheory
21	DP
22	DP
23	Graphs
24	StringMatching
25	NumberTheory
26	DP
27	DP
28	DP
29	Graphs
30	StringMatching
31	NumberTheory
32	DP
33	Graphs
34	DP
35	Graphs

36	StringMatching
37	NumberTheory
38	DP
39	Graphs
40	DP
41	Graphs
42	StringMatching
43	NumberTheory
44	DP
45	Graphs
46	DP
47	Graphs
48	StringMatching
49	NumberTheory
50	DP
51	Graphs
52	DP
53	Graphs
54	StringMatching
55	NumberTheory
56	DP
57	Graphs
58	DP
59	Graphs
60	StringMatching

11.3.8- ContestCreator:

```
INSERT INTO public."ContestCreator"(
        "creatorUsername", "assessmentScore")
        VALUES ('HussKiller', 90),
                ('ImanDestroyer', 90),
               ('BlueCoder42', 20),
```

('TechNinja91', 88),

('QuantumCoder7', 45),

('CodeMasterX', 29),

('PixelGeek23', 66),

('CyberPioneer', 78),

('SwiftSorcerer', 71),

('BinaryExplorer88', 86);

creatorUsername [PK] text	assessmentScore integer
BinaryExplorer88	86
BlueCoder42	20
CodeMasterX	29
CyberPioneer	78
HussKiller	90
ImanDestroyer	90
PixelGeek23	66
QuantumCoder7	45
SwiftSorcerer	71
TechNinja91	88

11.3.9- Organizes:

INSERT INTO public."Organizes"("creatorUsername", "roundNumber")

VALUES

('HussKiller', 1),

('HussKiller', 2),

('HussKiller', 3),

('HussKiller', 4),

('HussKiller', 5),

('ImanDestroyer', 6),

('ImanDestroyer', 7),

('ImanDestroyer', 8),

('ImanDestroyer', 9),

('ImanDestroyer', 10);

creatorUsername [PK] text	roundNumber [PK] integer
HussKiller	1
HussKiller	2
HussKiller	3
HussKiller	4
HussKiller	5
ImanDestroyer	6
ImanDestroyer	7
ImanDestroyer	8
ImanDestroyer	9
ImanDestroyer	10

11.3.10- Battle:

INSERT INTO public."Battle"("startTimestamp", "endTimestamp")

VALUES

('2023-01-15 14:00:00', '2023-01-15 16:00:00'),

('2023-02-05 13:30:00', '2023-02-05 15:30:00'),

('2023-03-20 11:45:00', '2023-03-20 13:45:00'),

('2023-04-10 09:15:00', '2023-04-10 11:15:00'),

('2023-05-02 10:00:00', '2023-05-02 12:00:00'),

('2023-06-15 08:30:00', '2023-06-15 10:30:00'),

('2023-07-08 14:45:00', '2023-07-08 16:45:00'),

('2023-08-19 12:15:00', '2023-08-19 14:15:00'),

('2023-09-25 13:00:00', '2023-09-25 15:00:00'),

('2023-10-15 11:30:00', '2023-10-15 13:30:00');

battleID [PK] integer	startTimestamp timestamp without time zone	endTimestamp timestamp without time zone
1	2023-01-15 14:00:00	2023-01-15 16:00:00
2	2023-02-05 13:30:00	2023-02-05 15:30:00
3	2023-03-20 11:45:00	2023-03-20 13:45:00
4	2023-04-10 09:15:00	2023-04-10 11:15:00
5	2023-05-02 10:00:00	2023-05-02 12:00:00
6	2023-06-15 08:30:00	2023-06-15 10:30:00
7	2023-07-08 14:45:00	2023-07-08 16:45:00
8	2023-08-19 12:15:00	2023-08-19 14:15:00
9	2023-09-25 13:00:00	2023-09-25 15:00:00
10	2023-10-15 11:30:00	2023-10-15 13:30:00

11.3.11- Joins:

INSERT INTO public."Joins"(username, "battleID", "isWinner")

VALUES

('Khaled47', 1, true),

('AliFarhat', 1, false),

('ay_charaf', 2, true),

('HussKiller', 2, false),

('ImanDestroyer', 3, true),

('Danosour', 3, false),

('HashTrash', 4, true),

('TheJoker', 4, false),

('MommyBoy', 5, true),

('cutiepie', 5, false),

('Tourist', 6, true),

('Khaled47', 6, false),

('MommyBoy', 7, true),

('HashTrash', 7, false),

('HashTrash', 8, true),

('Danosour', 8, false),

('ImanDestroyer', 9, true),

('HussKiller', 9, false),

('Tourist', 10, true),

('HashTrash', 10, false);

username [PK] text	battleID [PK] integer	isWinner boolean
AliFarhat	1	false
ay_charaf	2	true
cutiepie	5	false
Danosour	3	false
Danosour	8	false
HashTrash	4	true
HashTrash	7	false
HashTrash	8	true
HashTrash	10	false
HussKiller	2	false
HussKiller	9	false
ImanDestroyer	3	true
ImanDestroyer	9	true
Khaled47	1	true
Khaled47	6	false
MommyBoy	5	true
MommyBoy	7	true
TheJoker	4	false
Tourist	6	true
Tourist	10	true

11.3.12- Contestant:

```
INSERT INTO public."Contestant"("contestantUsername", rating)

VALUES

('Khaled47', 0),

('AliFarhat', 0),

('ay_charaf', 0),

('Danosour', 0),

('HashTrash', 0),
```

('TheJoker', 0),

('MommyBoy', 0),

('cutiepie', 0),

('Tourist', 0);

contestantUsername [PK] text	rating integer
AliFarhat	1750
ay_charaf	500
cutiepie	250
Danosour	699
HashTrash	500
Khaled47	500
MommyBoy	700
TheJoker	400
Tourist	1750

11.3.13- ConsistsOf:

INSERT INTO public."ConsistsOf"("battleID", "problemID")

VALUES

(1, 1),

(1, 2),

(1, 3),

(2, 7),

(2, 8),

(2, 9),

(3, 13),

(3, 14),

(3, 15),

(4, 19),

(4, 20),

(4, 21),

(5, 25),

(5, 26),

(5, 27),

(6, 31),

(6, 32),

(6, 33),

(7, 37),

(7, 38),

(7, 39),

(8, 43),

(8, 44),

(8, 45),

(9, 49),

(9, 50),

(9, 51),

(10, 55),

(10, 56),

(10, 57);

battleID [PK] integer	problemID [PK] integer
1	1
1	2
1	3
2	7
2	8
2	9
3	13
3	14
3	15
4	19
4	20
4	21
5	25
5	26
5	27
6	31
6	32
6	33
7	37
7	38
7	39
8	43
8	44
8	45
9	49
9	50
9	51
10	55
10	56
10	57

11.3.14- CompetesIn:

```
-- For user Khaled47

INSERT INTO public."CompetesIn"(

        "contestantUsername", "roundNumber", score, rank, "timestampOfRegistration")

        VALUES

        ('Khaled47', 1, 400, 1, '2023-01-10 08:30:00'), -- d

        ('Khaled47', 2, -100, 2, '2023-01-15 10:45:00'),-- d

        ('Khaled47', 3, 500, 3, '2023-01-22 13:15:00'), -- d

        ('Khaled47', 4, 500, 3, '2023-01-28 16:20:00'); -- d


-- For user AliFarhat

INSERT INTO public."CompetesIn"(

        "contestantUsername", "roundNumber", score, rank, "timestampOfRegistration")

        VALUES

        ('AliFarhat', 5, 500, 1, '2023-02-05 09:00:00'), --d

        ('AliFarhat', 6, 750, 1, '2023-02-10 11:45:00'), -- d

        ('AliFarhat', 7, -250, 3, '2023-02-16 14:30:00'), -- d

        ('AliFarhat', 8, 500, 1, '2023-02-22 16:45:00'); -- d


-- For user ay_charaf

INSERT INTO public."CompetesIn"(

        "contestantUsername", "roundNumber", score, rank, "timestampOfRegistration")

        VALUES
```

```
        ('ay_charaf', 9, 600, 1, '2023-03-05 08:15:00'), -- d
        ('ay_charaf', 10, 400, 2, '2023-03-10 09:30:00'), -- d
        ('ay_charaf', 1, 400, 2, '2023-03-16 12:45:00'), -- d
        ('ay_charaf', 2, -100, 3, '2023-03-22 14:00:00'); -- d

-- For user Danosour
INSERT INTO public."CompetesIn"(
        "contestantUsername", "roundNumber", score, rank, "timestampOfRegistration")
        VALUES
        ('Danosour', 3, 600, 2, '2023-04-05 09:30:00'), -- d
        ('Danosour', 4, 600, 2, '2023-04-10 11:15:00'), -- d
        ('Danosour', 5, 100, 4, '2023-04-16 14:30:00'), -- d
        ('Danosour', 6, -1, 4, '2023-04-22 16:45:00'); -- d

-- For user HashTrash
INSERT INTO public."CompetesIn"(
        "contestantUsername", "roundNumber", score, rank, "timestampOfRegistration")
        VALUES
        ('HashTrash', 7, 601, 1, '2023-05-05 08:45:00'), -- d
        ('HashTrash', 8, 300, 2, '2023-05-10 10:30:00'), -- d
        ('HashTrash', 9, 200, 3, '2023-05-16 13:15:00'), -- d
        ('HashTrash', 10, 200, 3, '2023-05-22 15:30:00'); -- d
-- For user TheJoker
```

```
INSERT INTO public."CompetesIn"(
        "contestantUsername", "roundNumber", score, rank, "timestampOfRegistration")
        VALUES
        ('TheJoker', 1, 200, 4, '2023-06-05 09:00:00'), -- d
        ('TheJoker', 2, 500, 1, '2023-06-10 11:15:00'), -- d
        ('TheJoker', 3, -100, 4, '2023-06-16 13:30:00'), -- d
        ('TheJoker', 4, 400, 4, '2023-06-22 15:45:00'); -- d

-- For user MommyBoy
INSERT INTO public."CompetesIn"(
        "contestantUsername", "roundNumber", score, rank, "timestampOfRegistration")
        VALUES
        ('MommyBoy', 5, 250, 3, '2023-07-05 08:30:00'), -- d
        ('MommyBoy', 6, 250, 3, '2023-07-10 10:45:00'), -- d
        ('MommyBoy', 7, -50, 2, '2023-07-16 13:00:00'), -- d
        ('MommyBoy', 8, 200, 3, '2023-07-22 15:15:00'); -- d

-- For user cutiepie
INSERT INTO public."CompetesIn"(
        "contestantUsername", "roundNumber", score, rank, "timestampOfRegistration")
        VALUES
        ('cutiepie', 9, 350, 2, '2023-08-05 08:45:00'), -- d
        ('cutiepie', 10, 400, 1, '2023-08-10 11:00:00'), -- d
```

```
        ('cutiepie', 1, 400, 3, '2023-08-16 13:15:00'), -- d
        ('cutiepie', 2, -100, 4, '2023-08-22 15:30:00'); -- d


-- For user Tourist
INSERT INTO public."CompetesIn"(
        "contestantUsername", "roundNumber", score, rank, "timestampOfRegistration")
        VALUES
        ('Tourist', 3, 1000, 1, '2023-09-05 09:00:00'), -- d
        ('Tourist', 4, 1000, 1, '2023-09-10 11:15:00'), -- d
        ('Tourist', 5, 350, 2, '2023-09-16 13:30:00'), -- d
        ('Tourist', 6, 400, 2, '2023-09-22 15:45:00'); -- d
```

contestantUsername [PK] text	roundNumber [PK] integer	score integer	rank integer	timestampOfRegistration timestamp without time zone
AliFarhat	5	500	1	2023-02-05 09:00:00
AliFarhat	6	750	1	2023-02-10 11:45:00
AliFarhat	8	500	1	2023-02-22 16:45:00
ay_charaf	2	-100	3	2023-03-22 14:00:00
ay_charaf	9	600	1	2023-03-05 08:15:00
cutiepie	2	-100	4	2023-08-22 15:30:00
cutiepie	9	350	2	2023-08-05 08:45:00
Danosour	3	570	2	2023-04-05 09:30:00
Danosour	5	70	4	2023-04-16 14:30:00
Danosour	6	-31	4	2023-04-22 16:45:00
HashTrash	8	300	2	2023-05-10 10:30:00
HashTrash	9	200	3	2023-05-16 13:15:00
Khaled47	2	-100	2	2023-01-15 10:45:00
Khaled47	3	500	3	2023-01-22 13:15:00
MommyBoy	5	250	3	2023-07-05 08:30:00
MommyBoy	6	250	3	2023-07-10 10:45:00
MommyBoy	8	200	3	2023-07-22 15:15:00
TheJoker	2	470	1	2023-06-10 11:15:00
TheJoker	3	-130	4	2023-06-16 13:30:00
Tourist	3	970	1	2023-09-05 09:00:00
Tourist	5	320	2	2023-09-16 13:30:00
Tourist	6	370	2	2023-09-22 15:45:00

11.3.15- Submission:

```
INSERT INTO public."Submission"(
        "programmingLanguage", "sourceCode", verdict, "timestamp")
        VALUES ('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 1 A
                ('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:33:00'), -- 2 B
                ('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:36:00'), -- 7 A
```

('GNU G++17 7.3.0', 'Code', 'Wrong Answer', '2023-01-14 09:30:00'), -- 8 B

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 13 A

('Java 11.0.6', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 14 B

('Java 11.0.6', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 15 C

('Java 11.0.6', 'Code', 'Wrong Answer', '2023-01-14 09:30:00'), -- 16 D

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 25 A

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 26 B

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 27 C

('GNU G++17 7.3.0', 'Code', 'Wrong Answer', '2023-01-14 09:30:00'), -- 28 D

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 31 A

('Java 11.0.6', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 32 B

('Java 11.0.6', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 33 C

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 37 A

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 38 B

('Python 3.8.10', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 39 C

('Python 3.8.10', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 40 D

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 43 A

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 44 B

('GNU G++17 7.3.0', 'Code', 'Wrong Answer', '2023-01-14 09:30:00'), -- 45 C

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 49 A

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 50 B

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 51 C

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 52 D

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 55 A

('Java 11.0.6', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 56 B

('Java 11.0.6', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 57 C

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 61 A

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 62 B

('GNU G++17 7.3.0', 'Code', 'Wrong Answer', '2023-01-14 09:30:00'), -- 63 C

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 1 A

('GNU G++17 7.3.0', 'Code', 'Wrong Answer', '2023-01-14 09:30:00'), -- 2 B

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 5 E

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 7 A

('GNU G++17 7.3.0', 'Code', 'Wrong Answer', '2023-01-14 09:30:00'), -- 12 F

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 13 A

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 14 B

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 15 C

('Java 11.0.6', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 25 A

('Java 11.0.6', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 26 B

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 27 C

('GNU G++17 7.3.0', 'Code', 'Wrong Answer', '2023-01-14 09:30:00'), -- 28 D

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 31 A

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 32 B

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 37 A

('GNU G++17 7.3.0', 'Code', 'Wrong Answer', '2023-01-14 09:30:00'), -- 38 B

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 43 A

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 44 B

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 45 C

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 49 A

('Python 3.8.10', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 52 D

('Python 3.8.10', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 57 C

('Python 3.8.10', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 61 A

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 62 B

('GNU G++17 7.3.0', 'Code', 'Wrong Answer', '2023-01-14 09:30:00'), -- 63 C

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 1 A

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 7 A

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 8 B

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 13 A

('GNU G++17 7.3.0', 'Code', 'Wrong Answer', '2023-01-14 09:30:00'), -- 14 B

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 25 A

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 26 B

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 27 C

('GNU G++17 7.3.0', 'Code', 'Wrong Answer', '2023-01-14 09:30:00'), -- 28 D

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 31 A

('GNU G++17 7.3.0', 'Code', 'Wrong Answer', '2023-01-14 09:30:00'), -- 32 B

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 37 A

('GNU G++17 7.3.0', 'Code', 'Wrong Answer', '2023-01-14 09:30:00'), -- 38 B

('Python 3.8.10', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 43 A

('Python 3.8.10', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 49 A

('Python 3.8.10', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 50 B

('GNU G++17 7.3.0', 'Code', 'Wrong Answer', '2023-01-14 09:30:00'), -- 51 C

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 55 A

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 61 A

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 62 B

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 1 A

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 2 B

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 3 C

('Python 3.8.10', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 7 A

('Python 3.8.10', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 8 B

('GNU G++17 7.3.0', 'Code', 'Wrong Answer', '2023-01-14 09:30:00'), -- 9 C

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 13 A

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 14 B

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 15 C

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 17 E

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 18 F

('Python 3.8.10', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 25 A

('Python 3.8.10', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 26 B

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 27 C

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 28 D

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 29 E

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 31 A

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 32 B

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 33 C

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 34 D

('Python 3.8.10', 'Code', 'Wrong Answer', '2023-01-14 09:30:00'), -- 35 E

('Python 3.8.10', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 37 A

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 38 B

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 39 C

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 40 D

('GNU G++17 7.3.0', 'Code', 'Accepted', '2023-01-14 09:30:00'), -- 41 E

('GNU G++17 7.3.0', 'Code', 'Wrong Answer', '2023-01-14 09:30:00'); -- 42 F

submissionID [PK] integer	programmingLanguage text	sourceCode text	verdict text	timestamp timestamp without time zone
1	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
2	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:33:00
3	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:36:00
4	GNU G++17 7.3.0	Code	Wrong Answer	2023-01-14 09:30:00
5	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
6	Java 11.0.6	Code	Accepted	2023-01-14 09:30:00
7	Java 11.0.6	Code	Accepted	2023-01-14 09:30:00
8	Java 11.0.6	Code	Wrong Answer	2023-01-14 09:30:00
9	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
10	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
11	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
12	GNU G++17 7.3.0	Code	Wrong Answer	2023-01-14 09:30:00
13	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
14	Java 11.0.6	Code	Accepted	2023-01-14 09:30:00
15	Java 11.0.6	Code	Accepted	2023-01-14 09:30:00
16	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
17	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
18	Python 3.8.10	Code	Accepted	2023-01-14 09:30:00
19	Python 3.8.10	Code	Accepted	2023-01-14 09:30:00
20	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
21	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
22	GNU G++17 7.3.0	Code	Wrong Answer	2023-01-14 09:30:00
23	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
24	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
25	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
26	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
27	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
28	Java 11.0.6	Code	Accepted	2023-01-14 09:30:00
29	Java 11.0.6	Code	Accepted	2023-01-14 09:30:00
30	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
31	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
32	GNU G++17 7.3.0	Code	Wrong Answer	2023-01-14 09:30:00
33	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
34	GNU G++17 7.3.0	Code	Wrong Answer	2023-01-14 09:30:00
35	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
36	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
37	GNU G++17 7.3.0	Code	Wrong Answer	2023-01-14 09:30:00
38	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
39	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
40	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00

41	Java 11.0.6	Code	Accepted	2023-01-14 09:30:00
42	Java 11.0.6	Code	Accepted	2023-01-14 09:30:00
43	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
44	GNU G++17 7.3.0	Code	Wrong Answer	2023-01-14 09:30:00
45	GNU G++17 7.3.0	Code	Hacked	2023-01-14 09:30:00
46	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
47	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
48	GNU G++17 7.3.0	Code	Wrong Answer	2023-01-14 09:30:00
49	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
50	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
51	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
52	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
53	Python 3.8.10	Code	Accepted	2023-01-14 09:30:00
54	Python 3.8.10	Code	Accepted	2023-01-14 09:30:00
55	Python 3.8.10	Code	Accepted	2023-01-14 09:30:00
56	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
57	GNU G++17 7.3.0	Code	Wrong Answer	2023-01-14 09:30:00
58	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
59	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
60	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
61	GNU G++17 7.3.0	Code	Hacked	2023-01-14 09:30:00
62	GNU G++17 7.3.0	Code	Wrong Answer	2023-01-14 09:30:00
63	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
64	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
65	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
66	GNU G++17 7.3.0	Code	Wrong Answer	2023-01-14 09:30:00
67	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
68	GNU G++17 7.3.0	Code	Wrong Answer	2023-01-14 09:30:00
69	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
70	GNU G++17 7.3.0	Code	Wrong Answer	2023-01-14 09:30:00
71	Python 3.8.10	Code	Accepted	2023-01-14 09:30:00
72	Python 3.8.10	Code	Accepted	2023-01-14 09:30:00
73	Python 3.8.10	Code	Accepted	2023-01-14 09:30:00
74	GNU G++17 7.3.0	Code	Wrong Answer	2023-01-14 09:30:00
75	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
76	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
77	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
78	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
79	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
80	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
81	Python 3.8.10	Code	Accepted	2023-01-14 09:30:00
82	Python 3.8.10	Code	Accepted	2023-01-14 09:30:00

83	GNU G++17 7.3.0	Code	Wrong Answer	2023-01-14 09:30:00
84	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
85	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
86	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
87	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
88	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
89	Python 3.8.10	Code	Accepted	2023-01-14 09:30:00
90	Python 3.8.10	Code	Accepted	2023-01-14 09:30:00
91	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
92	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
93	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
94	GNU G++17 7.3.0	Code	Hacked	2023-01-14 09:30:00
95	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
96	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
97	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
98	Python 3.8.10	Code	Wrong Answer	2023-01-14 09:30:00
99	Python 3.8.10	Code	Accepted	2023-01-14 09:30:00
100	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
101	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
102	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
103	GNU G++17 7.3.0	Code	Accepted	2023-01-14 09:30:00
104	GNU G++17 7.3.0	Code	Wrong Answer	2023-01-14 09:30:00

11.3.16- Submits:

```
INSERT INTO public."Submits"(
    username, "submissionID", "problemNumber")
VALUES ('Khaled47', 1, 1),
       ('Khaled47', 2, 2),
       ('Khaled47', 3, 7),
       ('Khaled47', 4, 8),
       ('Khaled47', 5, 13),
```

('Khaled47', 6, 14),

('Khaled47', 7, 15),

('Khaled47', 8, 16),

('Khaled47', 9, 19),

('Khaled47', 10, 20),

('Khaled47', 11, 21),

('Khaled47', 12, 22),

('AliFarhat', 13, 25),

('AliFarhat', 14, 26),

('AliFarhat', 15, 27),

('AliFarhat', 16, 31),

('AliFarhat', 17, 32),

('AliFarhat', 18, 33),

('AliFarhat', 19, 34),

('AliFarhat', 20, 37),

('AliFarhat', 21, 38),

('AliFarhat', 22, 39),

('AliFarhat', 23, 43),

('AliFarhat', 24, 44),

('AliFarhat', 25, 45),

('AliFarhat', 26, 46),

('ay_charaf', 27, 49),

('ay_charaf', 28, 50),

('ay_charaf', 29, 51),

('ay_charaf', 30, 55),

('ay_charaf', 31, 56),

('ay_charaf', 32, 57),

('ay_charaf', 33, 1),

('ay_charaf', 34, 2),

('ay_charaf', 35, 5),

('ay_charaf', 36, 7),

('ay_charaf', 37, 12),

('Danosour', 38, 13),

('Danosour', 39, 14),

('Danosour', 40, 15),

('Danosour', 41, 19),

('Danosour', 42, 20),

('Danosour', 43, 21),

('Danosour', 44, 22),

('Danosour', 45, 25),

('Danosour', 46, 26),

('Danosour', 47, 31),

('Danosour', 48, 32),

('HashTrash', 49, 37),

('HashTrash', 50, 38),

('HashTrash', 51, 39),

('HashTrash', 52, 43),

('HashTrash', 53, 36),

('HashTrash', 54, 51),

('HashTrash', 55, 55),

('HashTrash', 56, 56),

('HashTrash', 57, 57),

('TheJoker', 58, 1),

('TheJoker', 59, 7),

('TheJoker', 60, 8),

('TheJoker', 61, 13),

('TheJoker', 62, 14),

('TheJoker', 63, 19),

('TheJoker', 64, 20),

('TheJoker', 65, 22),

('TheJoker', 66, 22),

('MommyBoy', 67, 25),

('MommyBoy', 68, 26),

('MommyBoy', 69, 31),

('MommyBoy', 70, 32),

('MommyBoy', 71, 37),

('MommyBoy', 72, 43),

('MommyBoy', 73, 44),

('MommyBoy', 74, 45),

('cutiepie', 75, 49),

('cutiepie', 76, 55),

('cutiepie', 77, 56),

('cutiepie', 78, 1),

('cutiepie', 79, 2),

('cutiepie', 80, 3),

('cutiepie', 81, 7),

('cutiepie', 82, 8),

('cutiepie', 83, 9),

('Tourist', 84, 13),

('Tourist', 85, 14),

('Tourist', 86, 15),

('Tourist', 87, 17),

('Tourist', 88, 18),

('Tourist', 89, 19),

('Tourist', 90, 20),

('Tourist', 91, 21),

('Tourist', 92, 22),

('Tourist', 93, 23),

('Tourist', 94, 25),

('Tourist', 95, 26),

('Tourist', 96, 27),

('Tourist', 97, 28),

```
('Tourist', 98, 29),

('Tourist', 99, 31),

('Tourist', 100, 32),

('Tourist', 101, 33),

('Tourist', 102, 34),

('Tourist', 103, 35),

        ('Tourist', 104, 36);
```

username [PK] text	submissionID [PK] integer	problemNumber [PK] integer
AliFarhat	13	25
AliFarhat	14	26
AliFarhat	15	27
AliFarhat	16	31
AliFarhat	17	32
AliFarhat	18	33
AliFarhat	19	34
AliFarhat	20	37
AliFarhat	21	38
AliFarhat	22	39
AliFarhat	23	43
AliFarhat	24	44
AliFarhat	25	45
AliFarhat	26	46
ay_charaf	27	49
ay_charaf	28	50
ay_charaf	29	51
ay_charaf	30	55
ay_charaf	31	56
ay_charaf	32	57
ay_charaf	33	1
ay_charaf	34	2
ay_charaf	35	5
ay_charaf	36	7
ay_charaf	37	12
cutiepie	75	49
cutiepie	76	55
cutiepie	77	56
cutiepie	78	1
cutiepie	79	2
cutiepie	80	3
cutiepie	81	7
cutiepie	82	8
cutiepie	83	9
Danosour	38	13
Danosour	39	14
Danosour	40	15

Danosour	41	19
Danosour	42	20
Danosour	43	21
Danosour	44	22
Danosour	45	25
Danosour	46	26
Danosour	47	31
Danosour	48	32
HashTrash	49	37
HashTrash	50	38
HashTrash	51	39
HashTrash	52	43
HashTrash	53	46
HashTrash	54	51
HashTrash	55	55
HashTrash	56	56
HashTrash	57	57
Khaled47	1	1
Khaled47	2	2
Khaled47	3	7
Khaled47	4	8
Khaled47	5	13
Khaled47	6	14
Khaled47	7	15
Khaled47	8	16
Khaled47	9	19
Khaled47	10	20
Khaled47	11	21
Khaled47	12	22
MommyBoy	67	25
MommyBoy	68	26
MommyBoy	69	31
MommyBoy	70	32
MommyBoy	71	37
MommyBoy	72	43
MommyBoy	73	44
MommyBoy	74	45
TheJoker	58	1

TheJoker	59	7
TheJoker	60	8
TheJoker	61	13
TheJoker	62	14
TheJoker	63	19
TheJoker	64	20
TheJoker	65	21
TheJoker	66	22
Tourist	84	13
Tourist	85	14
Tourist	86	15
Tourist	87	17
Tourist	88	18
Tourist	89	19
Tourist	90	20
Tourist	91	21
Tourist	92	22
Tourist	93	23
Tourist	94	25
Tourist	95	26
Tourist	96	27
Tourist	97	28
Tourist	98	29
Tourist	99	31
Tourist	100	32
Tourist	101	33
Tourist	102	34
Tourist	103	35
Tourist	104	36

11.3.17- Messages:

```
INSERT INTO public."Messages"(
    "senderUsername", "receiverUsername", content, "timestamp")
VALUES
    ('AliFarhat', 'ay_charaf', 'Hey, lets see who can solve problem A first!', '2023-11-04 10:15:00'),
    ('ay_charaf', 'AliFarhat', 'Challenge accepted! Im already working on it!', '2023-11-05 14:22:00'),
    ('Khaled47', 'Danosour', 'Im the coding king! You dont stand a chance!', '2023-11-06 18:30:00'),
    ('Danosour', 'Khaled47', 'Well see about that. Prepare for the coding battle!', '2023-11-08 09:45:00'),
    ('HashTrash', 'TheJoker', 'You think youre a joker? Ill show you real coding humor!', '2023-11-10
16:50:00'),
    ('TheJoker', 'HashTrash', 'I always have a trick up my sleeve! Get ready!', '2023-11-12 22:35:00'),
    ('MommyBoy', 'cutiepie', 'Coding is childs play for me. Watch and learn!', '2023-11-15 11:20:00'),
    ('cutiepie', 'MommyBoy', 'A piece of cake, huh? Lets code our way through this!', '2023-11-17
13:45:00'),
    ('Tourist', 'Khaled47', 'Khaled47, youre just a tourist here. I rule!', '2023-11-19 17:25:00'),
    ('Khaled47', 'Tourist', 'Tourist, you may be the king, but Im the challenge!', '2023-11-21 08:30:00'),
    ('AliFarhat', 'ay_charaf', 'Ive solved problem A. It was easy!', '2023-11-23 09:55:00'),
    ('ay_charaf', 'AliFarhat', 'Well done! Now, lets tackle problem B!', '2023-11-26 14:40:00'),
    ('Danosour', 'HashTrash', 'The real showdown begins! Lets solve problem A.', '2023-11-28 15:05:00'),
    ('HashTrash', 'Danosour', 'Danosour, Im ready to take the challenge!', '2023-11-30 19:30:00'),
    ('TheJoker', 'MommyBoy', 'Ready for a coding joke? Here comes problem A.', '2023-12-01 22:55:00'),
    ('MommyBoy', 'TheJoker', 'I can handle your jokes! Lets tackle this challenge.', '2023-12-03 09:10:00'),
    ('cutiepie', 'Tourist', 'Tourist, Im right behind you. Problem A awaits!', '2023-12-05 12:20:00'),
```

('Tourist', 'cutiepie', 'Cutiepie, its a race! Lets solve it first.', '2023-12-07 15:40:00'),

('AliFarhat', 'Khaled47', 'Khaled47, you thought you could beat me? Problem A is done!', '2023-12-09 18:55:00'),

('Khaled47', 'AliFarhat', 'AliFarhat, Im always up for a challenge. Now, problem B awaits!', '2023-12-11 22:00:00');

senderUsername [PK] text	receiverUsername [PK] text	content text	timestamp [PK] timestamp without time zone
AliFarhat	ay_charaf	Hey, lets see who can solve problem A first!	2023-11-04 10:15:00
AliFarhat	ay_charaf	Ive solved problem A. It was easy!	2023-11-23 09:55:00
AliFarhat	Khaled47	Khaled47, you thought you could beat me? Problem A is done!	2023-12-09 18:55:00
ay_charaf	AliFarhat	Challenge accepted! Im already working on it!	2023-11-05 14:22:00
ay_charaf	AliFarhat	Well done! Now, lets tackle problem B!	2023-11-26 14:40:00
cutiepie	MommyBoy	A piece of cake, huh? Lets code our way through this!	2023-11-17 13:45:00
cutiepie	Tourist	Tourist, Im right behind you. Problem A awaits!	2023-12-05 12:20:00
Danosour	HashTrash	The real showdown begins! Lets solve problem A.	2023-11-28 15:05:00
Danosour	Khaled47	Well see about that. Prepare for the coding battle!	2023-11-08 09:45:00
HashTrash	Danosour	Danosour, Im ready to take the challenge!	2023-11-30 19:30:00
HashTrash	TheJoker	You think youre a joker? Ill show you real coding humor!	2023-11-10 16:50:00
Khaled47	AliFarhat	AliFarhat, Im always up for a challenge. Now, problem B awai...	2023-12-11 22:00:00
Khaled47	Danosour	Im the coding king! You dont stand a chance!	2023-11-06 18:30:00
Khaled47	Tourist	Tourist, you may be the king, but Im the challenge!	2023-11-21 08:30:00
MommyBoy	cutiepie	Coding is childs play for me. Watch and learn!	2023-11-15 11:20:00
MommyBoy	TheJoker	I can handle your jokes! Lets tackle this challenge.	2023-12-03 09:10:00
TheJoker	HashTrash	I always have a trick up my sleeve! Get ready!	2023-11-12 22:35:00
TheJoker	MommyBoy	Ready for a coding joke? Here comes problem A.	2023-12-01 22:55:00
Tourist	cutiepie	Cutiepie, its a race! Lets solve it first.	2023-12-07 15:40:00
Tourist	Khaled47	Khaled47, youre just a tourist here. I rule!	2023-11-19 17:25:00

11.3.18- Announcement:

INSERT INTO public."Announcement"(

"announcementID", "username", "language", "content", "timestamp")

VALUES

(1, 'HussKiller', 'English', 'Get ready for HusseinBakri Championship!', '2023-11-01 08:00:00'),

(2, 'HussKiller', 'English', 'Announcing AUB Showdown - Prepare for a coding battle!', '2023-11-03 10:30:00'),

(3, 'HussKiller', 'English', 'LAU Titans Challenge is coming up. Dont miss it!', '2023-11-05 12:15:00'),

(4, 'HussKiller', 'English', 'Its time for CodeCrafters Duel! Join the competition!', '2022-11-07 14:40:00'),

(5, 'HussKiller', 'English', 'Announcement: AlgorithmShip - Challenge your coding skills!', '2022-11-09 17:00:00'),

(6, 'ImanDestroyer', 'English', 'Rookie Coding Challenge is around the corner. Be ready!', '2021-11-11 19:25:00'),

(7, 'ImanDestroyer', 'English', 'Join CodeFusion Challenge and prove your coding expertise!', '2022-11-13 21:45:00'),

(8, 'ImanDestroyer', 'English', 'Get set for Algorithm Explorers - A coding competition like no other!', '2022-11-15 23:55:00'),

(9, 'ImanDestroyer', 'English', 'Announcing Dakowdas Enthusiasts - Show your coding skills!', '2022-11-17 08:10:00'),

(10, 'ImanDestroyer', 'English', 'Bakri Finale is coming soon. Get ready for the challenge!', '2023-9-19 10:30:00');

announcementID [PK] integer	username text	language text	content text	timestamp timestamp without time zone
1	HussKiller	English	Get ready for HusseinBakri Championship!	2023-11-01 08:00:00
2	HussKiller	English	Announcing AUB Showdown - Prepare for a coding battle!	2023-11-03 10:30:00
3	HussKiller	English	LAU Titans Challenge is coming up. Dont miss it!	2023-11-05 12:15:00
4	HussKiller	English	Its time for CodeCrafters Duel! Join the competition!	2022-11-07 14:40:00
5	HussKiller	English	Announcement: AlgorithmShip - Challenge your coding skills!	2022-11-09 17:00:00
6	ImanDestroyer	English	Rookie Coding Challenge is around the corner. Be ready!	2021-11-11 19:25:00
7	ImanDestroyer	English	Join CodeFusion Challenge and prove your coding expertise!	2022-11-13 21:45:00
8	ImanDestroyer	English	Get set for Algorithm Explorers - A coding competition like no oth...	2022-11-15 23:55:00
9	ImanDestroyer	English	Announcing Dakowdas Enthusiasts - Show your coding skills!	2022-11-17 08:10:00
10	ImanDestroyer	English	Bakri Finale is coming soon. Get ready for the challenge!	2023-09-19 10:30:00

11.3.19- Admin:

INSERT INTO public."Admin"(

```
        username, role, "contributionScore")
        VALUES ('MK', 'Super Admin', 5),
                ('TamTam', 'Moderator', 10),
               ('JohnSmith', 'Moderator', 15),
('EmmaJohnson', 'Super Admin', 8),
('MichaelWilson', 'Moderator', 12),
('SophiaJones', 'Moderator', 5),
('DavidBrown', 'Super Admin', 20),
('OliviaDavis', 'Moderator', 7),
('JamesMiller', 'Super Admin', 13),
('SophiaHarris', 'Moderator', 18);
```

username [PK] text	role text	contributionScore integer
DavidBrown	Super Admin	20
EmmaJohnson	Super Admin	8
JamesMiller	Super Admin	13
JohnSmith	Moderator	15
MichaelWilson	Moderator	12
MK	Super Admin	5
OliviaDavis	Moderator	7
SophiaHarris	Moderator	18
SophiaJones	Moderator	5
TamTam	Moderator	10

11.3.20- Responsibilities:

```
INSERT INTO public."Responsibilities"("adminUsername", responsibility)
VALUES ('JohnSmith', 'Manage User Accounts'),
```

```
    ('JohnSmith', 'Create Contests');
INSERT INTO public."Responsibilities"("adminUsername", responsibility)
VALUES ('EmmaJohnson', 'Review Contest Entries'),
    ('EmmaJohnson', 'Manage Announcements');
INSERT INTO public."Responsibilities"("adminUsername", responsibility)
VALUES ('MichaelWilson', 'Create Contests'),
    ('MichaelWilson', 'Monitor Chat Rooms');
INSERT INTO public."Responsibilities"("adminUsername", responsibility)
VALUES ('SophiaJones', 'Manage User Accounts'),
    ('SophiaJones', 'Manage Announcements');
INSERT INTO public."Responsibilities"("adminUsername", responsibility)
VALUES ('DavidBrown', 'Create Contests'),
    ('DavidBrown', 'Review Contest Entries');
INSERT INTO public."Responsibilities"("adminUsername", responsibility)
VALUES ('OliviaDavis', 'Monitor Chat Rooms'),
    ('OliviaDavis', 'Manage User Accounts');
INSERT INTO public."Responsibilities"("adminUsername", responsibility)
VALUES ('JamesMiller', 'Manage Announcements'),
    ('JamesMiller', 'Review Contest Entries');
INSERT INTO public."Responsibilities"("adminUsername", responsibility)
VALUES ('SophiaHarris', 'Create Contests'),
    ('SophiaHarris', 'Manage User Accounts');
INSERT INTO public."Responsibilities"("adminUsername", responsibility)
```

```
VALUES ('MK', 'Manage User Accounts'),
       ('MK', 'Create Contests'),
       ('MK', 'Moderate Forum');


INSERT INTO public."Responsibilities"("adminUsername", responsibility)
VALUES ('TamTam', 'Review Contest Entries'),
       ('TamTam', 'Monitor Chat Rooms'),
       ('TamTam', 'Manage Announcements');
```

adminUsername [PK] text	responsibility [PK] text
DavidBrown	Create Contests
DavidBrown	Review Contest Entries
EmmaJohnson	Manage Announceme...
EmmaJohnson	Review Contest Entries
JamesMiller	Manage Announceme...
JamesMiller	Review Contest Entries
JohnSmith	Create Contests
JohnSmith	Manage User Accounts
MichaelWilson	Create Contests
MichaelWilson	Monitor Chat Rooms
MK	Create Contests
MK	Manage User Accounts
MK	Moderate Forum
OliviaDavis	Manage User Accounts
OliviaDavis	Monitor Chat Rooms
SophiaHarris	Create Contests
SophiaHarris	Manage User Accounts
SophiaJones	Manage Announceme...
SophiaJones	Manage User Accounts
TamTam	Manage Announceme...
TamTam	Monitor Chat Rooms
TamTam	Review Contest Entries

11.3.21- BlogEntry:

INSERT INTO public."BlogEntry"(

username, title, content, "timestamp")

VALUES

('Khaled47', 'Some Sweet Tricks on Solving Dynamic Programming Problems', 'In this blog entry, Ill share some valuable insights and techniques for tackling dynamic programming problems. Dynamic programming can be a challenging topic, but with the right strategies, you can become a pro at solving such problems. Ill cover common patterns and provide clear examples to help you grasp the concepts better.', '2023-11-08 10:30:00'),

('HussKiller', 'My sweet set of problems for DSU', 'Welcome to my blog where Ive curated a collection of interesting problems related to Disjoint Set Union (DSU) data structure. DSU is a versatile tool for solving various computational problems. In this post, youll find a variety of problem statements, along with detailed explanations and code solutions to help you master DSU.', '2023-11-08 11:15:00'),

('ImanDestroyer', 'Tips and Tricks for increasing your RANK', 'Are you striving to improve your competitive programming rank? Look no further! In this blog, Ill share some practical tips and tricks that can help you enhance your competitive programming skills and boost your ranking. From efficient coding practices to effective time management, Ive got you covered.', '2023-11-08 12:00:00'),

('Danosour', 'Calculating Modulo Inverses Efficiently', 'Modulo inverses are a critical concept in number theory and cryptography. In this blog, Ill walk you through efficient techniques for calculating modulo inverses, providing clear explanations and code examples. Whether youre a beginner or an experienced programmer, youll find valuable insights here.', '2023-11-08 12:45:00'),

('TheJoker', 'Come Read Ma Blog on Solving Segment Tree Problems Yall', 'Hey there, folks! Join me in this exciting journey through solving problems using segment trees. Ill cover a range of segment tree-related challenges and provide step-by-step solutions. Get ready to become a segment tree pro with my guidance.', '2023-11-08 13:30:00'),

('Tourist', 'How I became top one on Dakowdas', 'As a competitive programmer, I know the struggles and joys of reaching the top ranks. In this blog, Ill share my personal story and the strategies I used to become a top competitor on platforms like Codeforces and AtCoder. I hope my experiences can inspire and guide you on your own journey to success.', '2023-11-08 14:15:00'),

('ay_charaf', 'Tips and Tricks on Solving Max Flow Problems', 'Max flow problems are classic in the world of algorithms. In this blog, Ill share valuable tips and tricks for solving max flow problems efficiently. From augmenting paths to the Ford-Fulkerson algorithm, youll find a comprehensive guide to mastering these problems.', '2023-11-08 15:00:00'),

('AliFarhat', 'Some Number Theory Ideas Used for Some Problems', 'Number theory is a fascinating branch of mathematics that finds applications in various algorithmic problems. In this blog, Ill introduce you to some number theory concepts and ideas that are frequently used in solving computational problems. Join me to explore the beauty of number theory and its practical relevance.', '2023-11-08 15:45:00'),

('HashTrash', 'Goin to Teach Ya Heree About Some Rollin Hashin Problems', 'Rolling hashing is a powerful technique for string matching and substring searches. In this blog, Ill teach you the fundamentals of rolling hashing and how to apply it to solve a variety of string-related problems. Get ready to level up your string manipulation skills!', '2023-11-08 16:30:00'),

('MK', 'Ideas on some Adjustments for Dakowdas', 'Competitive programming platforms like Codeforces are constantly evolving. In this blog, Ill share my thoughts and ideas on how you can adapt and make adjustments to stay competitive in the ever-changing landscape of competitive programming. Join the discussion and lets explore ways to thrive on these platforms.', '2023-11-08 17:15:00');

blogEntryID [PK] integer	username text	title text	content text	timestamp timestamp without time zone
1	Khaled47	Some Sweet Tricks on Solving Dynamic Programming Proble…	In this blog entry, Ill share some valuable insights and techniques for tackling dynamic programming problems. Dynamic progra…	2023-11-08 10:30:00
2	HussKiller	My sweet set of problems for DSU	Welcome to my blog where Ive curated a collection of interesting problems related to Disjoint Set Union (DSU) data structure. DS…	2023-11-08 11:15:00
3	ImanDestroyer	Tips and Tricks for increasing your RANK	Are you striving to improve your competitive programming rank? Look no further! In this blog, Ill share some practical tips and tri…	2023-11-08 12:00:00
4	Danosour	Calculating Modulo Inverses Efficiently	Modulo inverses are a critical concept in number theory and cryptography. In this blog, Ill walk you through efficient techniques f…	2023-11-08 12:45:00
5	TheJoker	Come Read Ma Blog on Solving Segment Tree Problems Yall	Hey there, folks! Join me in this exciting journey through solving problems using segment trees. Ill cover a range of segment tree-…	2023-11-08 13:30:00
6	Tourist	How I became top one on Dakowdas	As a competitive programmer, I know the struggles and joys of reaching the top ranks. In this blog, Ill share my personal story an…	2023-11-08 14:15:00
7	ay_charaf	Tips and Tricks on Solving Max Flow Problems	Max flow problems are classic in the world of algorithms. In this blog, Ill share valuable tips and tricks for solving max flow probl…	2023-11-08 15:00:00
8	AliFarhat	Some Number Theory Ideas Used for Some Problems	Number theory is a fascinating branch of mathematics that finds applications in various algorithmic problems. In this blog, Ill intr…	2023-11-08 15:45:00
9	HashTrash	Goin to Teach Ya Heree About Some Rollin Hashin Problems	Rolling hashing is a powerful technique for string matching and substring searches. In this blog, Ill teach you the fundamentals o…	2023-11-08 16:30:00
10	MK	Ideas on some Adjustments for Dakowdas	Competitive programming platforms like Codeforces are constantly evolving. In this blog, Ill share my thoughts and ideas on how…	2023-11-08 17:15:00

11.3.22- Comments:

INSERT INTO public."Comments"("blogEntryID", username, content, "timestamp")

VALUES (1, 'HussKiller', 'This blog is fantastic! The explanations are so clear and helpful.', '2023-11-08 10:45:00'),

(1, 'ImanDestroyer', 'I couldnt agree more! Its a great resource for DP problems.', '2023-11-08 11:00:00'),

(1, 'Tourist', 'Khaled47, youve nailed it with this blog. Keep up the good work!', '2023-11-08 11:15:00');

INSERT INTO public."Comments"("blogEntryID", username, content, "timestamp")

VALUES (2, 'Khaled47', 'HussKiller, your blog on DSU problems is a gem! Thanks for sharing.', '2023-11-08 11:30:00'),

(2, 'Tourist', 'Ive been looking for DSU problems, and this blog is a lifesaver.', '2023-11-08 11:45:00'),

(2, 'ImanDestroyer', 'This collection of DSU problems is a goldmine. Awesome work!', '2023-11-08 12:00:00');

INSERT INTO public."Comments"("blogEntryID", username, content, "timestamp")

VALUES (3, 'HussKiller', 'ImanDestroyer, your tips for improving rank are on point! Great job.', '2023-11-08 12:15:00'),

(3, 'Danosour', 'Ive already implemented some of these tips, and I see improvements in my ranking.', '2023-11-08 12:30:00'),

(3, 'TheJoker', 'I cant thank you enough for these valuable insights. Keep them coming!', '2023-11-08 12:45:00');

INSERT INTO public."Comments"("blogEntryID", username, content, "timestamp")

VALUES (4, 'TheJoker', 'Danosour, your explanation of modulo inverses is top-notch!', '2023-11-08 13:00:00'),

(4, 'HussKiller', 'Ive always struggled with modulo inverses, but your blog made it clear.', '2023-11-08 13:15:00'),

(4, 'Tourist', 'Thanks for simplifying this concept. Your blog is a game-changer.', '2023-11-08 13:30:00');

INSERT INTO public."Comments"("blogEntryID", username, content, "timestamp")

VALUES (5, 'ImanDestroyer', 'TheJoker, your segment tree blog is a must-read for anyone into data structures.', '2023-11-08 13:45:00'),

(5, 'Khaled47', 'Ive been struggling with segment trees, but your blog made it so much clearer.', '2023-11-08 14:00:00'),

(5, 'Danosour', 'Your explanations are spot on. Im learning a lot from your blog.', '2023-11-08 14:15:00');

INSERT INTO public."Comments"("blogEntryID", username, content, "timestamp")

VALUES (6, 'Khaled47', 'Tourist, your journey to the top is inspiring. Thanks for sharing your story.', '2023-11-08 14:30:00'),

(6, 'Danosour', 'Ive always looked up to top competitors like you. Your blog motivates me.', '2023-11-08 14:45:00'),

(6, 'ImanDestroyer', 'Your insights and strategies are invaluable. Keep reaching new heights!', '2023-11-08 15:00:00');

INSERT INTO public."Comments"("blogEntryID", username, content, "timestamp")

VALUES (7, 'ImanDestroyer', 'ay_charaf, your max flow blog is a comprehensive guide. Im learning a lot.', '2023-11-08 15:15:00'),

(7, 'HussKiller', 'Max flow problems used to intimidate me, but your blog is a game-changer.', '2023-11-08 15:30:00'),

(7, 'Tourist', 'I cant thank you enough for simplifying max flow problems. Great work!', '2023-11-08 15:45:00');

INSERT INTO public."Comments"("blogEntryID", username, content, "timestamp")

VALUES (8, 'HussKiller', 'AliFarhat, your number theory blog is a treasure trove of knowledge.', '2023-11-08 16:00:00'),

(8, 'Khaled47', 'Ive always been fascinated by number theory. Your blog satisfies my curiosity.', '2023-11-08 16:15:00'),

(8, 'TheJoker', 'Your explanations are so clear. Im becoming a number theory enthusiast!', '2023-11-08 16:30:00');

INSERT INTO public."Comments"("blogEntryID", username, content, "timestamp")

VALUES (9, 'TheJoker', 'HashTrash, your blog on rolling hashing is a gem. Ive learned so much.', '2023-11-08 16:45:00'),

(9, 'Tourist', 'Ive struggled with string matching problems, but your blog is a lifesaver.', '2023-11-08 17:00:00'),

(9, 'AliFarhat', 'Im getting better at string problems thanks to your guidance. Keep it up!', '2023-11-08 17:15:00');

INSERT INTO public."Comments"("blogEntryID", username, content, "timestamp")

VALUES (10, 'AliFarhat', 'MK, your ideas on adapting to competitive programming changes are spot on.', '2023-11-08 17:30:00'),

(10, 'TheJoker', 'Ive struggled to keep up with platform changes, but your blog is a roadmap.', '2023-11-08 17:45:00'),

(10, 'Tourist', 'Your insights are invaluable for staying competitive. Thanks for sharing!', '2023-11-08 18:00:00');

blogEntryID [PK] integer	username [PK] text	content text	timestamp timestamp without time zone
1	HussKiller	This blog is fantastic! The explanations are so clear and helpful.	2023-11-08 10:45:00
1	ImanDestroyer	I couldnt agree more! Its a great resource for DP problems.	2023-11-08 11:00:00
1	Tourist	Khaled47, youve nailed it with this blog. Keep up the good work!	2023-11-08 11:15:00
2	ImanDestroyer	This collection of DSU problems is a goldmine. Awesome work!	2023-11-08 12:00:00
2	Khaled47	HussKiller, your blog on DSU problems is a gem! Thanks for sharing.	2023-11-08 11:30:00
2	Tourist	Ive been looking for DSU problems, and this blog is a lifesaver.	2023-11-08 11:45:00
3	Danosour	Ive already implemented some of these tips, and I see improvements in my ranki...	2023-11-08 12:30:00
3	HussKiller	ImanDestroyer, your tips for improving rank are on point! Great job.	2023-11-08 12:15:00
3	TheJoker	I cant thank you enough for these valuable insights. Keep them coming!	2023-11-08 12:45:00
4	HussKiller	Ive always struggled with modulo inverses, but your blog made it clear.	2023-11-08 13:15:00
4	TheJoker	Danosour, your explanation of modulo inverses is top-notch!	2023-11-08 13:00:00
4	Tourist	Thanks for simplifying this concept. Your blog is a game-changer.	2023-11-08 13:30:00
5	Danosour	Your explanations are spot on. Im learning a lot from your blog.	2023-11-08 14:15:00
5	ImanDestroyer	TheJoker, your segment tree blog is a must-read for anyone into data structures.	2023-11-08 13:45:00
5	Khaled47	Ive been struggling with segment trees, but your blog made it so much clearer.	2023-11-08 14:00:00
6	Danosour	Ive always looked up to top competitors like you. Your blog motivates me.	2023-11-08 14:45:00
6	ImanDestroyer	Your insights and strategies are invaluable. Keep reaching new heights!	2023-11-08 15:00:00
6	Khaled47	Tourist, your journey to the top is inspiring. Thanks for sharing your story.	2023-11-08 14:30:00
7	HussKiller	Max flow problems used to intimidate me, but your blog is a game-changer.	2023-11-08 15:30:00
7	ImanDestroyer	ay_charaf, your max flow blog is a comprehensive guide. Im learning a lot.	2023-11-08 15:15:00
7	Tourist	I cant thank you enough for simplifying max flow problems. Great work!	2023-11-08 15:45:00
8	HussKiller	AliFarhat, your number theory blog is a treasure trove of knowledge.	2023-11-08 16:00:00
8	Khaled47	Ive always been fascinated by number theory. Your blog satisfies my curiosity.	2023-11-08 16:15:00
8	TheJoker	Your explanations are so clear. Im becoming a number theory enthusiast!	2023-11-08 16:30:00
9	AliFarhat	Im getting better at string problems thanks to your guidance. Keep it up!	2023-11-08 17:15:00
9	TheJoker	HashTrash, your blog on rolling hashing is a gem. Ive learned so much.	2023-11-08 16:45:00
9	Tourist	Ive struggled with string matching problems, but your blog is a lifesaver.	2023-11-08 17:00:00
10	AliFarhat	MK, your ideas on adapting to competitive programming changes are spot on.	2023-11-08 17:30:00
10	TheJoker	Ive struggled to keep up with platform changes, but your blog is a roadmap.	2023-11-08 17:45:00
10	Tourist	Your insights are invaluable for staying competitive. Thanks for sharing!	2023-11-08 18:00:00

11.3.23- Hack:

```
INSERT INTO public."Hack"(
        test, verdict, "timestamp")
        VALUES
        ('a n b', 'Hack Failed', '2023-11-08 10:30:00'),
        ('1 2 3','Hack Failed', '2023-11-08 11:15:00'),
        ('a n b','Hack Successful', '2023-11-08 12:00:00'),
```

('1 2 3','Hack Failed', '2023-11-08 12:45:00'),

('a n b','Hack Failed', '2023-11-08 13:30:00'),

('3 4939 2','Hack Successful', '2023-11-08 14:15:00'),

('afedfs','Hack Failed', '2023-11-08 15:00:00'),

('a 3 5','Hack Successful', '2023-11-08 15:45:00'),

('5 2 1','Hack Failed', '2023-11-08 16:30:00'),

('a n b','Hack Failed', '2023-11-08 17:15:00');

hackID [PK] integer	test text	verdict text	timestamp timestamp without time zone
1	a n b	Hack Failed	2023-11-08 10:30:00
2	1 2 3	Hack Failed	2023-11-08 11:15:00
3	a n b	Hack Successful	2023-11-08 12:00:00
4	1 2 3	Hack Failed	2023-11-08 12:45:00
5	a n b	Hack Failed	2023-11-08 13:30:00
6	3 4939 2	Hack Successful	2023-11-08 14:15:00
7	afedfs	Hack Failed	2023-11-08 15:00:00
8	a 3 5	Hack Successful	2023-11-08 15:45:00
9	5 2 1	Hack Failed	2023-11-08 16:30:00
10	a n b	Hack Failed	2023-11-08 17:15:00

11.3.24- Hacks:

INSERT INTO public."Hacks"(

"hackerUsername", "submissionID", "hackID")

VALUES ('Khaled47', 33, 1),

('Khaled47', 34, 2),

('AliFarhat', 45, 3),

('TheJoker', 13, 4),

('HashTrash', 20, 5),

('Danosour', 61, 6),

('Tourist', 67, 7),

('MommyBoy', 94, 8),

('ay_charaf', 54, 9),

('AliFarhat', 55, 10);

hackerUsername [PK] text	submissionID [PK] integer	hackID [PK] integer
AliFarhat	45	3
AliFarhat	55	10
ay_charaf	54	9
Danosour	61	6
HashTrash	20	5
Khaled47	33	1
Khaled47	34	2
MommyBoy	94	8
TheJoker	13	4
Tourist	67	7

11.3.25- AddsNotification:

INSERT INTO public."AddsNotification"(

"creatorUsername", "roundNumber", "timestamp", "notificationContent")

VALUES ('HussKiller', 2, '2023-02-05 10:40:00', 'Problem 1: Return value is log base 10 of a'),

('HussKiller', 2, '2023-02-05 10:45:00', 'Problem 1: a is different from b'),

('HussKiller', 2, '2023-02-05 10:58:00', 'Problem 2: length of c[] array is same as length of x[] array'),

('HussKiller', 1, '2023-01-15 08:10:00', 'Problem 2: Do not forget to do modulo'),

('HussKiller', 1, '2023-01-15 08:20:00', 'Problem 1: x values are inputed from user'),

('HussKiller', 1, '2023-01-15 09:00:00', 'Problem 1: No limit on x values'),

('ImanDestroyer', 6, '2023-06-15 11:00:00', 'Problem 3: Reminder to add 1 to final result for bias'),

('ImanDestroyer', 8,'2023-08-19 12:45:00', 'Problem1: X is gauranteed to be different from y'),

('ImanDestroyer', 9, '2023-09-25 10:45:00', 'Problem 4: y is concatination of z and w'),

('ImanDestroyer', 10, '2023-10-15 15:30:00', 'Problem 2: relation different from relationship');

creatorUsername [PK] text	roundNumber [PK] integer	timestamp [PK] timestamp without time zone	notificationContent text
HussKiller	1	2023-01-15 08:10:00	Problem 2: Do not forget to do modulo
HussKiller	1	2023-01-15 08:20:00	Problem 1: x values are inputed from user
HussKiller	1	2023-01-15 09:00:00	Problem 1: No limit on x values
HussKiller	2	2023-02-05 10:40:00	Problem 1: Return value is log base 10 of a
HussKiller	2	2023-02-05 10:45:00	Problem 1: a is different from b
HussKiller	2	2023-02-05 10:58:00	Problem 2: length of c[] array is same as length of x[] arr...
ImanDestroyer	6	2023-06-15 11:00:00	Problem 3: Reminder to add 1 to final result for bias
ImanDestroyer	8	2023-08-19 12:45:00	Problem1: X is gauranteed to be different from y
ImanDestroyer	9	2023-09-25 10:45:00	Problem 4: y is concatination of z and w
ImanDestroyer	10	2023-10-15 15:30:00	Problem 2: relation different from relationship

11.3.26- Hashtag:

INSERT INTO public."Hashtag"(

"blogEntryID", tag)

VALUES (1,'#CompetitiveProgramming'),

(1,'#Algorithms: '),

(1,'#CodingChallenge:'),

(1,'#DataStructures'),

(2,'#DynamicProgramming'),

(2,'#CodingTips'),

(3,'#ProgrammingContest'),

(3,'#ACM-ICPC'),

(4,'#AlgorithmDesign'),

(4,'#TechInterviews'),

(4,'#DevLife'),

(5,'#CodeChallengeAccepted'),

(5,'#DebuggingSkills'),

(5,'#CodeSnippets'),

(5,'#CodeDebate');

blogEntryID [PK] integer	tag [PK] text
1	#Algorithms:
1	#CodingChallenge:
1	#CompetitiveProgramming
1	#DataStructures
2	#CodingTips
2	#DynamicProgramming
3	#ACM-ICPC
3	#ProgrammingContest
4	#AlgorithmDesign
4	#DevLife
4	#TechInterviews
5	#CodeChallengeAccepted
5	#CodeDebate
5	#CodeSnippets
5	#DebuggingSkills

11.3.27- VotesOn:

VotesOn:

```
INSERT INTO public."VotesOn"(
        "blogEntryID", username)
        VALUES (1, 'Khaled47'),
(1, 'AliFarhat'),
(1, 'HussKiller'),
(1, 'ImanDestroyer'),
(1, 'Danosour'),
(2, 'Khaled47'),
(2, 'AliFarhat'),
(2, 'Danosour'),
(2, 'ay_charaf'),
(2, 'HussKiller'),
(6, 'HashTrash'),
(6, 'Danosour'),
(6, 'AliFarhat'),
(6, 'ay_charaf'),
(6, 'ImanDestroyer'),
(6, 'TheJoker'),
(6, 'MommyBoy'),
(8, 'ay_charaf'),
(8, 'cutiepie'),
(8, 'HussKiller'),
(8, 'Tourist');
```

blogEntryID [PK] integer	username [PK] text
1	AliFarhat
1	Danosour
1	HussKiller
1	ImanDestroyer
1	Khaled47
2	AliFarhat
2	ay_charaf
2	Danosour
2	HussKiller
2	Khaled47
6	AliFarhat
6	ay_charaf
6	Danosour
6	HashTrash
6	ImanDestroyer
6	MommyBoy
6	TheJoker
8	ay_charaf
8	cutiepie
8	HussKiller
8	Tourist

11.4-Views

The following section is for creating views in order to display derived attributes.

11.4.1- View for NumberOfFriends & NumberOfProblemsSolved Derived Attributes in User

```
CREATE VIEW User_View AS

SELECT *,
```

(Select COUNT(*) from public."Befriends" b where ("friend1Username" = u."username")or("friend2Username" = u."username")) as NumberOfFriends,

(Select COUNT(DISTINCT p."problemID")

from public."ProgrammingProblem" p

join public."Submits" ss on (ss."problemNumber"=p."problemID" and ss."username"=u."username")

join public."Submission" s on ss."submissionID"=s."submissionID"

where (s."verdict"='Accepted')) as NumberOfProblemsSolved

from public."User" u ;

```
CREATE VIEW User_View AS
SELECT *,
(Select COUNT(*) from public."Befriends" b where ("friend1Username" = u."username")or("friend2Username" = u."username")) as NumberOfFriends,
(Select COUNT(DISTINCT p."problemID")
 from public."ProgrammingProblem" p
 join public."Submits" ss on (ss."problemNumber"=p."problemID" and ss."username"=u."username")
 join public."Submission" s on ss."submissionID"=s."submissionID"
 where (s."verdict"='Accepted')) as NumberOfProblemsSolved
from public."User" u ;
```

username text	email text	passwod text	firstName text	lastName text	country text	registrationDate date	numberoffriends bigint	numberofproblemssolved bigint
Khaled47	Khaled47@gmail.com	5e884898da28047151d0e56f8dc6292773603d0d6b8b4b9b883883a1541d3…	Khaled	Charaf	Lebanon	2023-01-15	4	9
AliFarhat	AliFarhat@gmail.com	2d3e1312b4f10a8a7c0ebc7dc5ea8cf2cbca99e7c8bd6a57f188e7e1e5571d7	Ali	Farhat	Lebanon	2022-10-21	5	13
ay_charaf	ay_charaf@gmail.com	3a63c03f43710ad8eae4c0d6de315df0d73f0b3b5f5ae8e09d4da598b968c66	Ayman	Charaf	Lebanon	2023-06-08	4	8
HussKiller	HussKiller@gmail.com	ad8b3b7d8755165555a80f8b0b5f196ea44d6c0a54f491b7a2a66e8dfdece0b	Hussein	Bakri	Lebanon	2022-07-30	5	0
ImanDestroyer	ImanDestroyer@gmail.com	ea4109f401e9aabbab4c5fb17c25f31e5c64bfa5e1efcdfcf4c9ec9910b896d	Iman	Ghalayini	Lebanon	2023-03-04	6	0
Danosour	Danosour@gmail.com	1f548a2d8992b4d8a6cb424f3c6aeef19e719e5d158c6fb39e9ea2ae23f4570	Dana	Kossaybeyati	Lebanon	2022-08-16	3	8
HashTrash	HashTrash@gmail.com	f5297c5e46c2f95e5c700f946e77dd715583cfb41895fcb52e4d85b7eb38127	Hashem	Khodor	Lebanon	2023-02-14	4	8
TheJoker	TheJoker@gmail.com	efbc56e0d798f9372831ceab35f8c3d812ead0c4e1c1ac21f96c0f8bb4249f7	Omar	Ramadan	Lebanon	2023-02-04	6	6
MommyBoy	MommyBoy@gmail.com	fd1fc301decbde0a5ad5c4c27ae398d07bb9ec75c7e2c1b11a76d0068d15cd	Abdelatif	Saleh	Lebanon	2023-01-29	2	5
cutiepie	cutiepie@gmail.com	e2bf8da1bde8ed5e6ccf3f936da984ae11a2c0403c54f64eb69a1e7b1d3d62c	Lama	Hasbini	America	2023-02-21	1	8
Tourist	Tourist@gmail.com	5e884898da28047151d0e56f8dc6292773603d0d6b8b4b9b883883a1541d3…	Gennady	Korotkevich	Belarus	2022-08-16	0	18
MK	MK@gmail.com	6600bf6331c98f1619619fc8e405d5ec3dd97e614b9d241f14275168f130c916	Mohamad	Kreidieh	Lebanon	2023-11-10	0	0
TamTam	TamTam@gmail.com	6c3bf86f56a3e67d88f6d6bea7e03dd5ca3c31b07a51fc1ce28571e049dce5ea	Tamara	Sadek	Lebanon	2023-11-10	0	0
JohnSmith	JohnSmith@gmail.com	d19edeb7198f1e9ee2ef8d1876f49b3ec3c4c3fc126c6419734397741f048d3…	John	Smith	USA	2023-11-10	0	0
EmmaJohnson	EmmaJohnson@gmail.com	f01c86dbff968e105264b2dab1b71f0631a36047aa4b0f66a4b51b69cb68ff12	Emma	Johnson	USA	2023-11-10	0	0
MichaelWilson	MichaelWilson@outlook.com	bfc47af6e83f5a3f98f3c5ec42022100b453bd5d6d17c79d05d7482d309558d7	Michael	Wilson	Canada	2023-11-10	0	0
SophiaJones	SophiaJones@mail.com	8e11a7fbf6d88abf9c20758bb8c005855542f6c2edc283f8d8d5d9eb61f4c5e6	Sophia	Jones	Canada	2023-11-10	0	0
DavidBrown	DavidBrown@mail.com	0bf10ed62bf8762c753ab8c1b9d9326b53c7cd0491deea8db3f3a6b675400b…	David	Brown	UK	2023-11-10	0	0
OliviaDavis	OliviaDavis@yahoo.com	e6ccf9d13c1167c3a6b61c0176473cd977640999ef974ea83a75be88918046…	Olivia	Davis	UK	2023-11-10	0	0
JamesMiller	JamesMiller@gmail.com	2a0a3842a5046e301f3e4d24cfb2d5fb924932f6c3c72318d5f1cfa216e68549	James	Miller	Australia	2023-11-10	0	0
SophiaHarris	SophiaHarris@gmail.com	cfdad2c74ab40a22fc08c67183ad02ce431826b26d69c650ff8c877ed489b950	Sophia	Harris	Australia	2023-11-10	0	0
BlueCoder42	bluecoder@gmail.com	YSBN93u28y382	Alex	Turner	Japan	2023-04-15	0	0
TechNinja91	techtech@mail.com	sUb292372804	Maya	Rodriguez	Brazil	2022-09-28	0	0
QuantumCoder7	quantumcoder@mail.com	482t40	Ethan	Bennett	Australia	2023-07-03	0	0
CodeMasterX	masterrr23@gmail.com	suhsYV27	Olivia	Chang	Germany	2022-11-22	0	0
PixelGeek23	pixy77@mail.com	826HVGT2	Liam	Mitchell	India	2020-05-18	0	0
CyberPioneer	cyber101@gmail.com	dhYGF27	Ava	Thompson	Canada	2020-08-09	0	0
SwiftSorcerer	swiftt1@mail.com	7S28B2G	Noah	Patel	South Africa	2022-12-30	0	0
BinaryExplorer88	binbin77@mail.com	SSS8oj1b	Mia	Sanchez	Lebanon	2019-01-10	0	0

11.4.2- View for TimeElapsedSinceAnnouncement Derived Attribute in Announcement

CREATE VIEW Announcement_View AS

Select *, AGE(NOW(), "timestamp") AS Time_Elapsed_Since_Announcement

from public."Announcement";

```
CREATE VIEW Announcement_View AS
Select *, AGE(NOW(), "timestamp") AS Time_Elapsed_Since_Announcement
from public."Announcement";
```

announcementID integer	username text	language text	content text	timestamp timestamp without time zone	time_elapsed_since_announcement interval
1	HussKiller	English	Get ready for HusseinBakri Championship!	2023-11-01 08:00:00	9 days 08:02:06.032633
2	HussKiller	English	Announcing AUB Showdown - Prepare for a coding battle!	2023-11-03 10:30:00	7 days 05:32:06.032633
3	HussKiller	English	LAU Titans Challenge is coming up. Dont miss it!	2023-11-05 12:15:00	5 days 03:47:06.032633
4	HussKiller	English	Its time for CodeCrafters Duel! Join the competition!	2022-11-07 14:40:00	1 year 3 days 01:22:06.032633
5	HussKiller	English	Announcement: AlgorithmShip - Challenge your coding skills!	2022-11-09 17:00:00	1 year 23:02:06.032633
6	ImanDestroyer	English	Rookie Coding Challenge is around the corner. Be ready!	2021-11-11 19:25:00	1 year 11 mons 28 days 20:37:06.032633
7	ImanDestroyer	English	Join CodeFusion Challenge and prove your coding expertise!	2022-11-13 21:45:00	11 mons 26 days 18:17:06.032633
8	ImanDestroyer	English	Get set for Algorithm Explorers - A coding competition like no oth…	2022-11-15 23:55:00	11 mons 24 days 16:07:06.032633
9	ImanDestroyer	English	Announcing Dakowdas Enthusiasts - Show your coding skills!	2022-11-17 08:10:00	11 mons 23 days 07:52:06.032633
10	ImanDestroyer	English	Bakri Finale is coming soon. Get ready for the challenge!	2023-09-19 10:30:00	1 mon 21 days 05:32:06.032633

11.4.3- View for Votes Derived Attribute in BlogEntry

CREATE VIEW BlogEntry_View AS

SELECT *,

(Select Count(*) from public."VotesOn" v where v."blogEntryID" = b."blogEntryID") AS votes

from public."BlogEntry" b

```
CREATE VIEW BlogEntry_View AS
SELECT *,
(Select Count(*) from public."VotesOn" v where v."blogEntryID" = b."blogEntryID") AS votes
from public."BlogEntry" b
```

blogEntryID integer	username text	title text	content text	timestamp timestamp without time zone	votes bigint
1	Khaled47	Some Sweet Tricks on Solving Dynamic Programming Proble...	In this blo...	2023-11-08 10:30:00	5
2	HussKiller	My sweet set of problems for DSU	Welcome ...	2023-11-08 11:15:00	5
3	ImanDestroyer	Tips and Tricks for increasing your RANK	Are you st...	2023-11-08 12:00:00	0
4	Danosour	Calculating Modulo Inverses Efficiently	Modulo in...	2023-11-08 12:45:00	0
5	TheJoker	Come Read Ma Blog on Solving Segment Tree Problems Yall	Hey there,...	2023-11-08 13:30:00	0
6	Tourist	How I became top one on Dakowdas	As a com...	2023-11-08 14:15:00	7
7	ay_charaf	Tips and Tricks on Solving Max Flow Problems	Max flow ...	2023-11-08 15:00:00	0
8	AliFarhat	Some Number Theory Ideas Used for Some Problems	Number t...	2023-11-08 15:45:00	4
9	HashTrash	Goin to Teach Ya Heree About Some Rollin Hashin Problems	Rolling ha...	2023-11-08 16:30:00	0
10	MK	Ideas on some Adjustments for Dakowdas	Competiti...	2023-11-08 17:15:00	0

11.4.4- View for NumberOfCorrectSubmissions Derived Attribute in ProgrammingProblem

CREATE VIEW ProgrammingProblem_View AS

SELECT *,

(SELECT COUNT(*) from public."Submission" s join public."Submits" ss on ss."submissionID" = s."submissionID" where (ss."problemNumber" = p."problemID" and s."verdict" = 'Accepted')) as number_of_correct_submissions

from public."ProgrammingProblem" p;

```
CREATE VIEW ProgrammingProblem_View AS
SELECT *,
(SELECT COUNT(*) from public."Submission" s
    join public."Submits" ss on ss."submissionID" = s."submissionID"
    where (ss."problemNumber" = p."problemID" and s."verdict" = 'Accepted')) as number_of_correct_submissions
 from public."ProgrammingProblem" p;
```

problemID integer	difficultyLevel character	title text	description text	timeLimit integer	memoryLimit integer	roundNumber integer	number_of_correct_submissions bigint
1	A	Two Sum Problem	Given an array of integers and a target sum, find tw…	2	256	1	4
2	B	Palindrome Check	Write a program to check if a given string is a palin…	3	512	1	2
3	C	Maximum Subarray	Find the contiguous subarray with the largest sum i…	4	128	1	1
4	D	Factorial Calculation	Write a program to calculate the factorial of a given…	5	256	1	0
5	E	Graph Traversal	Traverse a graph and perform depth-first or breadth…	1	512	1	1
6	F	0/1 Knapsack Problem	Solve the 0/1 Knapsack problem using dynamic pro…	4	256	1	0
7	A	Greatest Common Divisor	Find the greatest common divisor (GCD) of two pos…	1	128	2	4
8	B	Binary Tree Height	Calculate the height of a binary tree, which is the le…	2	512	2	2
9	C	Longest Common Substring	Find the longest common substring between two gi…	3	256	2	0
10	D	Sparse Matrix Multiplication	Perform multiplication of two sparse matrices effic…	4	128	2	0
11	E	Shortest Path in Maze	Find the shortest path from the start to the end in a…	5	512	2	0
12	F	Matrix Chain Multiplication	Solve the matrix chain multiplication problem using…	1	256	2	0
13	A	Rotated Sorted Array Search	Search for a target element in a rotated sorted arra…	2	256	3	3
14	B	Binary Tree Traversal	Perform various traversals (inorder, preorder, posto…	3	512	3	3
15	C	Regular Expression Matcher	Implement a regular expression matcher that can …	4	128	3	3
16	D	Sudoku Solver	Solve a Sudoku puzzle by filling in the empty cells …	5	256	3	0
17	E	Chess Game Logic	Implement the logic for a basic chess game, includi…	1	512	3	1
18	F	Longest Common Subsequence	Find the longest common subsequence of two give…	4	256	3	1
19	A	Longest Increasing Subsequence	Find the length of the longest increasing subseque…	1	128	4	4
20	B	Shortest Path	Implement Dijkstras algorithm to find the shortest …	2	512	4	4
21	C	Knuth-Morris-Pratt	Write an algorithm to search for a pattern in a text …	3	256	4	4
22	D	Prime Factorization	Factorize a given integer into its prime factors and t…	4	128	4	1
23	E	Maximum Subarray Sum	Find the maximum sum of a contiguous subarray w…	5	512	4	1
24	F	Traveling Salesman	Solve the Traveling Salesman Problem using dyna…	1	256	4	0
25	A	Longest Palindromic Substring	Find the longest palindromic substring in a given st…	1	128	5	2
26	B	Minimum Spanning Tree	Implement Kruskals algorithm to find the minimum …	2	512	5	3
27	C	Boyer-Moore	Write an algorithm to search for a pattern in a text …	3	256	5	2
28	D	Modular Exponentiation	Compute large powers modulo a prime number effi…	4	128	5	1
29	E	Coin Change	Find the number of ways to make change for a give…	5	512	5	0
30	F	Longest Common Subarray	Find the longest common subarray between two arr…	1	256	5	0
31	A	Maximum Subarray Sum (Non-Contiguous)	Find the maximum sum of a non-contiguous subarr…	2	256	6	4
32	B	Topological Sorting	Implement topological sorting for directed acyclic …	3	512	6	2
33	C	Rabin-Karp	Write an algorithm to search for a pattern in a text …	4	256	6	2
34	D	Greatest Common Divisor	Find the greatest common divisor (GCD) of two inte…	5	128	6	2
35	E	Longest Increasing Subsequence	Find the length of the longest increasing subseque…	1	512	6	1
36	F	Bipartite Graph Check	Determine if a given graph is bipartite or not using …	2	256	6	0
37	A	Matrix Chain Multiplication	Solve the matrix chain multiplication problem using…	3	128	7	3
38	B	Strongly Connected Components	Implement Tarjans algorithm to find strongly conne…	4	512	7	2
39	C	Aho-Corasick	Write an algorithm to search for multiple patterns i…	5	256	7	1
40	D	Sieve of Eratosthenes	Generate prime numbers up to a given limit using t…	1	128	7	0
41	E	Longest Common Subsequence	Find the longest common subsequence of two seq…	2	512	7	0
42	F	Shortest Path (Negative Weight)	Implement the Bellman-Ford algorithm to find the s…	3	256	7	0

43	A	Coin Change (Minimum Coins)	Find the minimum number of coins needed to make...	4	128	8	3
44	B	Articulation Points and Bridges	Find articulation points and bridges in an undirecte...	5	512	8	2
45	C	Longest Common Prefix Array	Compute the longest common prefix array for a set...	1	256	8	1
46	D	Modular Multiplicative Inverse	Calculate the modular multiplicative inverse of an i...	2	128	8	2
47	E	Rod Cutting	Solve the rod cutting problem to maximize profit us...	3	512	8	0
48	F	Maximum Flow	Find the maximum flow in a flow network using the...	4	256	8	0
49	A	Longest Palindromic Subsequence	Find the length of the longest palindromic subsequ...	5	128	9	2
50	B	Minimum Spanning Tree (Prims Algorithm)	Implement Prims algorithm to find the minimum sp...	1	512	9	1
51	C	Z Algorithm	Compute the Z array to efficiently search for a patt...	2	256	9	2
52	D	Chinese Remainder Theorem	Solve simultaneous modular congruences using th...	3	128	9	0
53	E	Edit Distance	Find the edit distance (Levenshtein distance) betwe...	4	512	9	0
54	F	Hamiltonian Path	Find a Hamiltonian path in a directed or undirected ...	5	256	9	0
55	A	Longest Zigzag Subsequence	Find the length of the longest zigzag subsequence i...	1	128	10	3
56	B	Maximum Bipartite Matching	Find the maximum cardinality matching in a bipartit...	2	512	10	3
57	C	Suffix Array	Construct a suffix array for a given string to enable ...	3	256	10	0
58	D	Lucas Theorem	Apply Lucass Theorem to compute binomial coeffi...	4	128	10	0
59	E	Longest Increasing Subarray	Find the longest increasing subarray within an array...	5	512	10	0
60	F	Maximum Planar Subgraph	Find the maximum planar subgraph of a planar gra...	1	256	10	0

11.4.5- View for TimeBeforeStart Derived Attribute in Contest

CREATE VIEW Contest_View AS

SELECT *,AGE("startTimestamp", NOW()) AS time_before_start

from public."Contest";

```
CREATE VIEW Contest_View AS
SELECT *,AGE("startTimestamp", NOW()) AS time_before_start
from public."Contest";
```

roundNumber integer	name text	divsion integer	startTimestamp timestamp without time zone	length time without time zone	description text	time_before_start interval
1	HusseinBakri Championship	1	2023-01-15 08:00:00	03:00:00	The ultimate coding challenge for elite programmers.	-9 mons -26 days -08:24:49.198051
2	AUB Showdown	2	2023-02-05 10:30:00	02:30:00	Test your algorithmic skills in this exciting contest.	-9 mons -5 days -05:54:49.198051
3	LAU Titans Challenge	3	2023-03-20 09:15:00	03:00:00	An opportunity for newcomers to prove their tech prowess.	-7 mons -21 days -07:09:49.198051
4	CodeCrafters Duel	1	2023-04-10 14:00:00	04:00:00	A battle of wits among the coding elite.	-7 mons -02:24:49.198051
5	AlgorithmShip	2	2023-05-02 11:45:00	02:30:00	Marvel at the magic of algorithms in this contest and get a Scholarsh...	-6 mons -8 days -04:39:49.198051
6	Rookie Coding Challenge	3	2023-06-15 10:00:00	03:00:00	An ideal platform for budding programmers to shine.	-4 mons -25 days -06:24:49.198051
7	CodeFusion Challenge	1	2023-07-08 13:30:00	04:00:00	Merge your coding skills in this fusion of challenges.	-4 mons -2 days -02:54:49.198051
8	Algorithm Explorers	2	2023-08-19 11:15:00	02:30:00	Explore the depths of algorithms in this exciting contest.	-2 mons -22 days -05:09:49.198051
9	Dakowdas Enthusiasts	3	2023-09-25 09:45:00	03:00:00	A showdown for tech enthusiasts to conquer.	-1 mons -15 days -06:39:49.198051
10	Bakri Finale	1	2023-10-15 14:15:00	04:00:00	Place top 1 to get a 100 on Bakris database course at AUB!	-26 days -02:09:49.198051

11.4.6- View for Bracket Derived Attribute in Contestant

Step 1: rating_to_bracket

Function that translates a contestant's rating into the corresponding bracket.

```
CREATE FUNCTION rating_to_bracket(rating integer)
RETURNS text language plpgsql AS $$
BEGIN
        if (rating <= 1900) then
                return 'Newbie';
        elsif (rating <=1399) then
                return 'Pupil';
        elsif (rating <=1599) then
                return 'Specialist';
        elsif (rating <= 1899) then
                return 'Expert';
        elsif (rating <= 2099) then
                return 'Candidate Master';
        elsif (rating <=2299) then
                return 'Master';
        elsif (rating <= 2399) then
                return 'International Master';
        elsif (rating <= 2599) then
                return 'Grandmaster';
        elsif (rating <= 2999) then
                return 'International Grandmaster';
        else
                return 'Legendary Grandmaster';
```

```
    END IF;
END;
$$;
```

```
SELECT rating_to_bracket(2500)
```

Output Messages Notifications

rating_to_bracket text
Grandmaster

Step 2: View

```
CREATE VIEW Contestant_View AS
SELECT *, rating_to_bracket("rating") as bracket
from public."Contestant"
```

```
CREATE VIEW Contestant_View AS
SELECT *, rating_to_bracket("rating") as bracket
from public."Contestant"
```

contestantUsername text	rating integer	bracket text
Khaled47	500	Newbie
ay_charaf	500	Newbie
Danosour	699	Newbie
HashTrash	500	Newbie
TheJoker	400	Newbie
MommyBoy	700	Newbie
cutiepie	250	Newbie
Tourist	1750	Newbie
AliFarhat	1950	Candidate Master

11.5-Useful SQL Queries

The following section is for implementing some purposeful queries that are vital for other queries and the development phase.

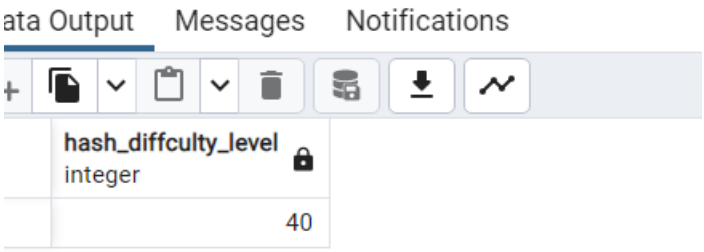
11.5.1- Hashing Difficulty Level

Programming problems of difficulty levels A, B, C, D, E, or F are worth 30, 40, 50, 60, 70, and 80 points respectively. This is used to update the scores.

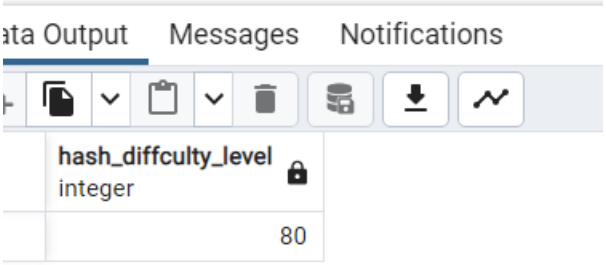
```
CREATE OR REPLACE FUNCTION hash_diffculty_level(difficultyLevel character)

RETURNS integer AS $$

BEGIN

        IF (difficultyLevel = 'A') THEN

                return 30;

        ELSIF (difficultyLevel = 'B') THEN

                return 40;

        ELSIF (difficultyLevel = 'C') THEN

                return 50;

        ELSIF (difficultyLevel = 'D') THEN

                return 60;

        ELSIF (difficultyLevel = 'E') THEN

                return 70;

        ELSE

                return 80;

        END IF;

END;
```

$$ LANGUAGE plpgsql;

```
SELECT hash_diffculty_level('B')
```




```
SELECT hash_diffculty_level('F')
```



11.5.2- Stored Procedure For Rating Update

This stored procedure to update the ratings of contestants upon a change in their scores. It's called by the function hack_score_update() defined in section 11.6.

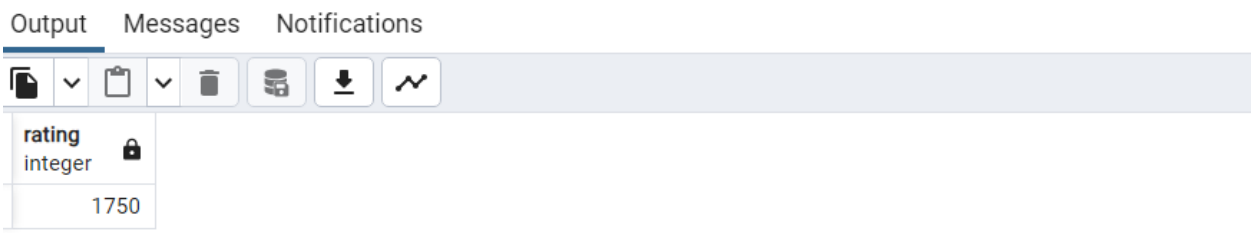
```
CREATE OR REPLACE PROCEDURE update_rating(usern text) AS $$
BEGIN
        UPDATE public."Contestant"
        set "rating" = GREATEST ((Select sum(c."score") from public."CompetesIn" as c where c."contestantUsername"=usern),0);
END;
$$ LANGUAGE plpgsql;
```

11.5.3- Trigger for Updating in Rating After Competition

When a contestant competes in a contest, his/her rating get updates based on the score that he/she achieved in the contest. If the score they attained renders their rating negative, then their rating is set to 0.

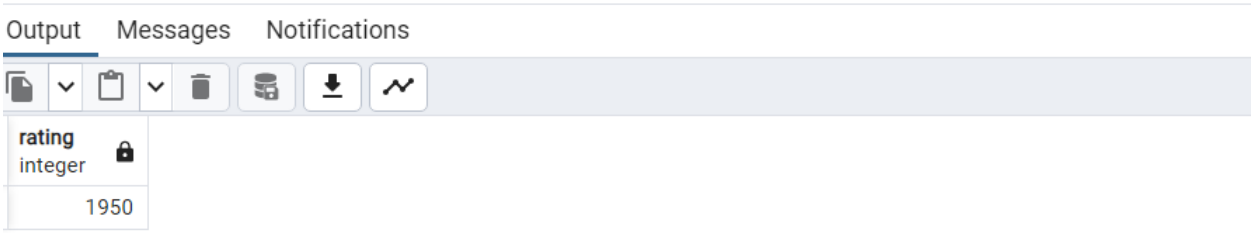
```
CREATE OR REPLACE FUNCTION rating_update_for_1_contest()

RETURNS TRIGGER AS $$

BEGIN

        UPDATE public."Contestant"

        SET "rating" = GREATEST(0, "rating" + NEW."score")

        where "contestantUsername" = NEW."contestantUsername";


        RETURN NEW;

END;


$$ LANGUAGE plpgsql;


CREATE TRIGGER new_competition_trigger

AFTER INSERT

ON public."CompetesIn"

FOR EACH ROW

EXECUTE FUNCTION rating_update_for_1_contest();
```

```
SELECT "rating" from public."Contestant" where "contestantUsername" = 'AliFarhat'
```




```
INSERT INTO public."CompetesIn"(
    "contestantUsername", "roundNumber", score, rank, "timestampOfRegistration")
    VALUES ('AliFarhat', 9, 200, 4, '2023-09-25 09:00:00' );
```

```
SELECT "rating" from public."Contestant" where "contestantUsername" = 'AliFarhat'
```



11.5.4-Selecting Top Scorers In a Contest

The following query selects the 3 top-scoring contestants in round number 3. Note that round Number may be changed in order to select the 3 top-scoring contestants in any round. The limit may also be changed to select the top 1,2,4,etc contestants.

```
Select *
from public."CompetesIn"
where "roundNumber" = 3
Order By "score" DESC
LIMIT 3;
```

```sql
Select *
from public."CompetesIn"
where "roundNumber" = 3
Order By "score" DESC
LIMIT 3;
```

Output Messages Notifications

contestantUsername [PK] text	roundNumber [PK] integer	score integer	rank integer	timestampOfRegistration timestamp without time zone
Tourist	3	970	1	2023-09-05 09:00:00
Danosour	3	570	2	2023-04-05 09:30:00
Khaled47	3	500	3	2023-01-22 13:15:00

11.5.5- Selecting Top Blog Entries on a Certain Topic

Given a string which refers to some programming topic, list the top 5 blog entries with the highest number of votes related to this topic (get that from the hashtags and number of votes). Display these blog entries with their corresponding publisher's name, with it's blog title, it's content, and each number of votes

```sql
CREATE OR REPLACE FUNCTION search_topic(topic text)

RETURNS TABLE (username text , title text, _content text, _timestamp timestamp, votes bigint)

language plpgsql  AS $$

BEGIN

        RETURN QUERY

                Select v."username", v."title", v."content", v."timestamp", v."votes"

                from BlogEntry_View as v

                join public."Hashtag" as h on h."blogEntryID" = v."blogEntryID"

                where h."tag" = topic

                order by "votes" DESC
```

```
        Limit 5;

END;

$$;
```

```
INSERT INTO public."Hashtag"(
    "blogEntryID", tag)
    VALUES (2, '#DataStructures'),
           (3, '#DataStructures'),
           (6, '#DataStructures'),
           (8, '#DataStructures'),
           (10, '#DataStructures');
```

```
select * from search_topic('#DataStructures')
```

Output Messages Notifications

username text	title text	_content text	_timestamp timestamp without time zone	votes bigint
Tourist	How I became top one on Dakowdas	As a competitive programmer, I know the stru...	2023-11-08 14:15:00	7
Khaled47	Some Sweet Tricks on Solving Dynamic Programming Proble...	In this blog entry, Ill share some valuable insig...	2023-11-08 10:30:00	5
HussKiller	My sweet set of problems for DSU	Welcome to my blog where Ive curated a colle...	2023-11-08 11:15:00	5
AliFarhat	Some Number Theory Ideas Used for Some Problems	Number theory is a fascinating branch of mat...	2023-11-08 15:45:00	4
ImanDestroyer	Tips and Tricks for increasing your RANK	Are you striving to improve your competitive p...	2023-11-08 12:00:00	0

11.6- Complex SQL Queries

11.6.1- Trigger for Successful Hack

Yikes! Your submission has been hacked. Unfortunately, your submission is no longer valid, and your score will be deducted. The hacker on the other hand earned 50 points for his diligent work.

```
CREATE OR REPLACE FUNCTION hack_score_update()

RETURNS TRIGGER AS $$

DECLARE

   contestant_username text;
```

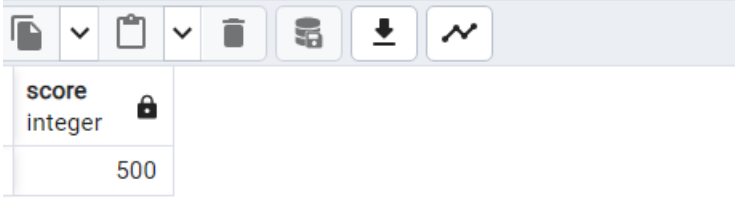
```
BEGIN
    IF ((SELECT "verdict" FROM public."Hack" WHERE "hackID" = NEW."hackID") = 'Hack Successful') THEN
        UPDATE public."Submission"
        SET verdict = 'Hacked'
        WHERE "submissionID" = NEW."submissionID";

        SELECT c."contestantUsername"
        INTO contestant_username
        FROM public."Contestant" c
        JOIN public."User" u ON u."username" = c."contestantUsername"
        JOIN public."Submits" s ON s."username" = u."username"
        WHERE s."submissionID" = NEW."submissionID";

        CALL update_rating(contestant_username);

        UPDATE public."CompetesIn"
        SET score = score - hash_diffculty_level(
            (SELECT p."difficultyLevel"
             FROM public."ProgrammingProblem" p
             JOIN public."Submits" st ON p."problemID" = st."problemNumber"
             WHERE st."submissionID" = NEW."submissionID")
        )
        WHERE ("contestantUsername" = contestant_username)
```

```
        AND ("roundNumber" = (
            SELECT p."roundNumber"
            FROM public."ProgrammingProblem" p
            JOIN public."Submits" st ON p."problemID" = st."problemNumber"
            WHERE st."submissionID" = NEW."submissionID"
        ));

    UPDATE public."CompetesIn"
    SET score = score + 50
    WHERE ("contestantUsername" = NEW."hackerUsername")
        AND ("roundNumber" = (
            SELECT p."roundNumber"
            FROM public."ProgrammingProblem" p
            JOIN public."Submits" st ON p."problemID" = st."problemNumber"
            WHERE st."submissionID" = NEW."submissionID"
        ));

    CALL update_rating(NEW."hackerUsername");
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Example: Here, Danosour hacks AliFarhat's submission number 13 in round number 5.

```sql
Select "score"
from public."CompetesIn"
where "contestantUsername" = 'AliFarhat'
and "roundNumber" = 5;
```

Output Messages Notifications



score integer
500

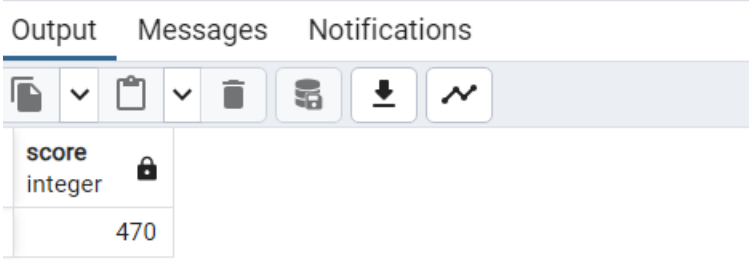
```sql
Select "score"
from public."CompetesIn"
where "contestantUsername" = 'Danosour'
and "roundNumber" = 5;
```

Output Messages Notifications

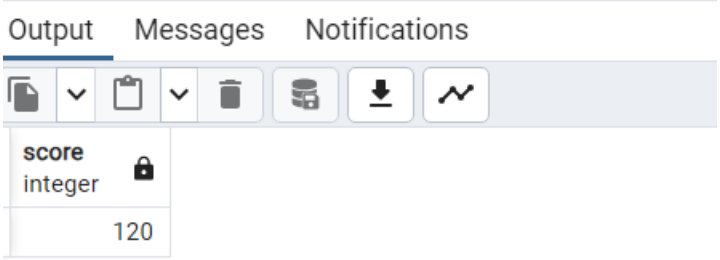
score integer
70

```sql
INSERT INTO public."Hack"(
    test, verdict, "timestamp")
    VALUES ('aa a', 'Hack Successful', '2023-05-02 10:00:00');

INSERT INTO public."Hacks"(
    "hackerUsername", "submissionID", "hackID")
    VALUES ('Danosour', 13, 11);
```

```
Select "score"
from public."CompetesIn"
where "contestantUsername" = 'AliFarhat'
and "roundNumber" = 5;
```




```
Select "score"
from public."CompetesIn"
where "contestantUsername" = 'Danosour'
and "roundNumber" = 5;
```



11.6.2- Viewing Friend's Scores in Contest

A user can view his friends' performance in the contests that he competes in.

```
CREATE OR REPLACE FUNCTION view_friends_in_competition (contestantusername text, round integer)

RETURNS TABLE (frinedusername text, friendscore integer, friendrank integer) language plpgsql  AS $$

DECLARE

        friend_record RECORD;
```

```
BEGIN

        IF EXISTS(SELECT * FROM public."CompetesIn" c where c."contestantUsername" =
contestantusername and c."roundNumber" = round) then

                FOR friend_record in

                        SELECT "contestantUsername", "score" , "rank"

                                from public."CompetesIn"

                                where (("contestantUsername" =

                                --CASE 1: contestantusername is friend1

                                ((Select b."friend2Username" from public."Befriends" b

                                                join public."User" u on
b."friend2Username" = u."username"

                                                join public."Contestant" c on
c."contestantUsername" = u."username"

                                                join public."CompetesIn" ci on
c."contestantUsername" = ci."contestantUsername"

                                                where
b."friend1Username"=contestantusername and ci."roundNumber" = round)

                                UNION

                                 --CASE 2: contestantusername is friend2

                                 (Select b."friend1Username" from public."Befriends" b

                                                join public."User" u on
b."friend1Username" = u."username"

                                                join public."Contestant" c on
c."contestantUsername" = u."username"
```

```
                                                    join public."CompetesIn" ci on
c."contestantUsername" = ci."contestantUsername"
                                                    where
b."friend2Username"=contestantusername and ci."roundNumber" = round)
                                        )       )
                                        and
                                        "roundNumber" = round
                                        )
                                        ORDER BY "score" DESC
            LOOP
                frinedUsername:=friend_record."contestantUsername";
                friendScore:=friend_record."score";
                friendRank:=friend_record."rank";
                RETURN NEXT;
            END LOOP;
        END IF;
END;
$$;
```

```
CREATE OR REPLACE FUNCTION view_friends_in_competition (contestantusername text, round integer)
RETURNS TABLE (frinedusername text, friendscore integer, friendrank integer) language plpgsql  AS $$
DECLARE
    friend_record RECORD;
BEGIN
    IF EXISTS(SELECT * FROM public."CompetesIn" c where c."contestantUsername" = contestantusername and c."roundNumber" = round) then
        FOR friend_record in
            SELECT "contestantUsername", "score" , "rank"
                    from public."CompetesIn"
                    where (("contestantUsername" =
                    --CASE 1: contestantusername is friend1
                    ((Select b."friend2Username" from public."Befriends" b
                                join public."User" u on b."friend2Username" = u."username"
                                join public."Contestant" c on c."contestantUsername" = u."username"
                                join public."CompetesIn" ci on c."contestantUsername" = ci."contestantUsername"
                                where b."friend1Username"=contestantusername and ci."roundNumber" = round)
                    UNION
                     --CASE 2: contestantusername is friend2
                     (Select b."friend1Username" from public."Befriends" b
                                join public."User" u on b."friend1Username" = u."username"
                                join public."Contestant" c on c."contestantUsername" = u."username"
                                join public."CompetesIn" ci on c."contestantUsername" = ci."contestantUsername"
                                where b."friend2Username"=contestantusername and ci."roundNumber" = round)
                     )  )
                     and
                     "roundNumber" = round
                     )
                     ORDER BY "score" DESC
        LOOP
            frinedUsername:=friend_record."contestantUsername";
            friendScore:=friend_record."score";
            friendRank:=friend_record."rank";
            RETURN NEXT;
        END LOOP;
    END IF;
END;
$$;
```

```
Select * from view_friends_in_competition('Danosour',3)
```

Output Messages Notifications

frinedusername text	friendscore integer	friendrank integer
TheJoker	-130	4

11.6.3- Active Users

Dakowdaz.com defines an active user as someone who has engaged with the platform in various ways, such as participating in at least one battle, posting at least one blog, or taking part in at least one contest within the past year. The website owners aim to assess the count of active users, focusing on individuals who have demonstrated recent activity on the platform through these specified actions.

```
select count(u."username") as numberOfActiveUsers
        from public."User" as u
        where u."username" in (
                                                SELECT DISTINCT (u.username)
                                                        FROM public."User" as u
                                                        join public."Joins" as j on u."username"
= j."username"
                                                        join public."Battle" as b on b."battleID"
= j."battleID"
                                                        where Extract(year from
(b."startTimestamp" - current_timestamp)) = 0
                                                UNION(
                                                        SELECT DISTINCT (u.username)
                                                                FROM public."User" as u
                                                                join public."BlogEntry" as b on
u."username" = b."username"
                                                                where Extract(year from
(b."timestamp" - current_timestamp)) = 0
                                                        )
                                                        UNION(
                                                                select DISTINCT
(c."contestantUsername")
                                                                from public."Contestant" as c
                                                                join public."CompetesIn" as c2
on c."contestantUsername" = c2."contestantUsername"
                                                                join public."Contest" as c3 on
c2."roundNumber" = c3."roundNumber"
```

where Extract(year from
(c3."startTimestamp" - current_timestamp)) = 0
)
)

```
select count(u."username") as numberOfActiveUsers
    from public."User" as u
    where u."username" in (
                            SELECT DISTINCT (u.username)
                                FROM public."User" as u
                                join public."Joins" as j on u."username" = j."username"
                                join public."Battle" as b on b."battleID" = j."battleID"
                                where Extract(year from (b."startTimestamp" - current_timestamp)) = 0
                            UNION(
                                SELECT DISTINCT (u.username)
                                    FROM public."User" as u
                                    join public."BlogEntry" as b on u."username" = b."username"
                                    where Extract(year from (b."timestamp" - current_timestamp)) = 0
                                )
                                UNION(
                                    select DISTINCT (c."contestantUsername")
                                    from public."Contestant" as c
                                    join public."CompetesIn" as c2 on c."contestantUsername" = c2."contestantUsername"
                                    join public."Contest" as c3 on c2."roundNumber" = c3."roundNumber"
                                    where Extract(year from (c3."startTimestamp" - current_timestamp)) = 0
                                    )
                                )
```

11.6.4- International Championship

At dakowdaz.com, we are organizing an international championship, and we want to invite the highest-rated user from each country to participate. To achieve this, we have crafted a SQL query that identifies the top-rated contestant in each country. The query selects distinct combinations of rating, username, and country using the DISTINCT ON clause, ensuring that only the user with the highest rating from each country is included in the result set. The ordering is done by country and rating in descending order, prioritizing the top-rated contestants. This query enables us to efficiently gather participants for the country world championship based on their exceptional ratings.

```
SELECT DISTINCT ON (u."country") u."country", c.rating, u."username"
FROM public."User" u
JOIN public."Contestant" c ON u."username" = c."contestantUsername"
WHERE c.rating = (
   SELECT MAX(c2.rating)
   FROM public."Contestant" c2
   JOIN public."User" u2 ON c2."contestantUsername" = u2."username"
   WHERE u2."country" = u."country"
)
ORDER BY u."country", c.rating DESC;
```

```
SELECT DISTINCT ON (u."country") u."country", c.rating, u."username"
FROM public."User" u
JOIN public."Contestant" c ON u."username" = c."contestantUsername"
WHERE c.rating = (
    SELECT MAX(c2.rating)
    FROM public."Contestant" c2
    JOIN public."User" u2 ON c2."contestantUsername" = u2."username"
    WHERE u2."country" = u."country"
)
ORDER BY u."country", c.rating DESC;
```

Output Messages Notifications

country text	rating integer	username text
America	250	cutiepie
Belarus	1750	Tourist
Lebanon	1950	AliFarhat

11.6.5- User Interaction Analysis:

Dakowdaz.com is interested in understanding user interactions. They want a report that shows the total number of comments, blog entries, and messages each user has made. Additionally, they want to know the total number of friendships each user has established.

```
SELECT
    u.username,
    nOfCommentsSubquery.numberOfComments as numberOfComments,
    nOfBlogEntriesSubquery.numberOfBlogEntriesPosted as numberOfBlogEntriesPosted,
    numberOfMessagesSubquery.numberOfMessages as numberOfMessages,
    numberOfFriendsSubquery.numberOfFriends as numberOfFriends
FROM
    public."User" as u
LEFT JOIN
    (SELECT u2.username, count(u2.username) as numberOfComments
    FROM public."User" u2
    JOIN public."Comments" c ON u2."username" = c."username"
    GROUP BY u2.username) as nOfCommentsSubquery
    ON nOfCommentsSubquery."username" = u."username"
LEFT JOIN
    (SELECT u.username, count(u.username) as numberOfBlogEntriesPosted
    FROM public."User" u
    JOIN public."BlogEntry" b ON u."username" = b."username"
    GROUP BY u.username) as nOfBlogEntriesSubquery
    ON nOfBlogEntriesSubquery."username" = u."username"
LEFT JOIN
    (SELECT u.username, count(u.username) as numberOfMessages
    FROM public."User" u
    JOIN public."Messages" m ON u."username" = m."senderUsername"
    GROUP BY u.username) as numberOfMessagesSubquery
    ON numberOfMessagesSubquery."username" = u."username"
LEFT JOIN
    (SELECT u.username, count(u.username) as numberOfFriends
    FROM public."User" u
    JOIN public."Befriends" bf ON u."username" = bf."friend1Username"
    GROUP BY u.username) as numberOfFriendsSubquery
```

```
ON numberOfFriendsSubquery."username" = u."username";
```

username [PK] text	numberofcomments bigint	numberofblogentriesposted bigint	numberofmessages bigint	numberoffriends bigint
Khaled47	4	1	3	4
AliFarhat	2	1	3	4
ay_charaf	0	1	2	2
HussKiller	5	1	0	2
ImanDestroyer	5	1	0	2
Danosour	3	1	2	2
HashTrash	0	1	2	2
TheJoker	5	1	2	2
MommyBoy	0	0	2	0
cutiepie	0	0	2	0
Tourist	6	1	2	0
MK	0	1	0	0
TamTam	0	0	0	0
JohnSmith	0	0	0	0
EmmaJohnson	0	0	0	0
MichaelWilson	0	0	0	0
SophiaJones	0	0	0	0
DavidBrown	0	0	0	0
OliviaDavis	0	0	0	0
JamesMiller	0	0	0	0
SophiaHarris	0	0	0	0

11.6.6- Awarding Contest Creators:

The team at Dakowdaz.com has truly excelled in organizing contests this year! We are eager to recognize and reward the exceptional efforts of the top three contest creators who have contributed significantly to the success of these events. We're keen on obtaining detailed information about these top-ranking contest creators to ensure that their dedication and creativity are appropriately acknowledged with well-deserved awards.

```
WITH creator_ranking AS (
  SELECT
    "creatorUsername",
    RANK() OVER (ORDER BY "assessmentScore" DESC) as ranking
  FROM public."ContestCreator"
```

)

SELECT

u."username",

u."email",

u."firstName",

u."lastName",

r."ranking"

FROM creator_ranking as r

JOIN public."User" as u on u."username" = r."creatorUsername"

WHERE ranking <= 3

ORDER BY ranking

```
WITH creator_ranking AS (
  SELECT
    "creatorUsername",
    RANK() OVER (ORDER BY "assessmentScore" DESC) as ranking
  FROM public."ContestCreator"
)
SELECT
  u."username",
  u."email",
  u."firstName",
  u."lastName",
  r."ranking"
FROM creator_ranking as r
JOIN public."User" as u on u."username" = r."creatorUsername"
WHERE ranking <= 3
ORDER BY ranking
```

Output Messages Notifications

username [PK] text	email text	firstName text	lastName text	ranking bigint
ImanDestroyer	ImanDestroyer@gmail.com	Iman	Ghalayini	1
HussKiller	HussKiller@gmail.com	Hussein	Bakri	1
TechNinja91	techtech@mail.com	Maya	Rodriguez	3

11.6.7- Leading Users

Dakowdaz.com wants to build a leaderboard to showcase the coding prowess of its users. We need to identify the top 5 users based on the count of distinct problems they have successfully solved by utilizing a recursive approach.

```
WITH RECURSIVE UserProblems AS (
    SELECT DISTINCT u."username", s."submissionID", pp."difficultyLevel"
    FROM public."Submits" s
    JOIN public."User" u ON s."username" = u."username"
    JOIN public."Submission" ss ON s."submissionID" = ss."submissionID"
    JOIN public."ProgrammingProblem" pp ON s."problemNumber" = pp."problemID"
    WHERE ss."verdict" = 'Accepted'

    UNION

    SELECT up."username", s."submissionID", pp."difficultyLevel"
    FROM UserProblems up
    JOIN public."Submits" s ON up."submissionID" = s."submissionID"
    JOIN public."Submission" ss ON s."submissionID" = ss."submissionID"
    JOIN public."ProgrammingProblem" pp ON s."problemNumber" = pp."problemID"
    WHERE ss."verdict" = 'Accepted'
)

SELECT "username", COUNT(DISTINCT "submissionID") AS solved_problems_count
FROM UserProblems
GROUP BY "username"
ORDER BY "solved_problems_count" DESC
LIMIT 5;
```

```sql
WITH RECURSIVE UserProblems AS (
    SELECT DISTINCT u."username", s."submissionID", pp."difficultyLevel"
    FROM public."Submits" s
    JOIN public."User" u ON s."username" = u."username"
    JOIN public."Submission" ss ON s."submissionID" = ss."submissionID"
    JOIN public."ProgrammingProblem" pp ON s."problemNumber" = pp."problemID"
    WHERE ss."verdict" = 'Accepted'

    UNION

    SELECT up."username", s."submissionID", pp."difficultyLevel"
    FROM UserProblems up
    JOIN public."Submits" s ON up."submissionID" = s."submissionID"
    JOIN public."Submission" ss ON s."submissionID" = ss."submissionID"
    JOIN public."ProgrammingProblem" pp ON s."problemNumber" = pp."problemID"
    WHERE ss."verdict" = 'Accepted'
)

SELECT "username", COUNT(DISTINCT "submissionID") AS solved_problems_count
FROM UserProblems
GROUP BY "username"
ORDER BY "solved_problems_count" DESC
LIMIT 5;
```

Data Output | Messages | Notifications

	username text	solved_problems_count bigint
1	Tourist	18
2	AliFarhat	12
3	Khaled47	9
4	Danosour	8
5	HashTrash	8

11.6.8- Champion Showcase: Unveiling Stellar Performers in Dakowdaz Coding Contest Round

Dakowdaz.com intends to present the performance of participants in a particular contest round (round 1), highlighting their submissions for each of the six specified problems (A to F). The goal is to display the users who successfully solved each problem, and the results will be ordered from the highest to the lowest score for clarity and ease of understanding.

```sql
SELECT
   u."contestantUsername",
   -- Subquery A
   EXISTS (
      SELECT 1
      FROM public."Submits" as s1
      JOIN public."Submission" as s2 ON s2."submissionID" = s1."submissionID"
      WHERE s2."verdict" = 'Accepted' AND s1."username" = u."contestantUsername" AND
s1."problemNumber" = 1
   ) as A,
   -- Subquery B
   EXISTS (
      SELECT 1
      FROM public."Submits" as s3
```

```
        JOIN public."Submission" as s4 ON s4."submissionID" = s3."submissionID"
        WHERE s4."verdict" = 'Accepted' AND s3."username" = u."contestantUsername" AND
s3."problemNumber" = 2
    ) as B,
    -- Subquery C
    EXISTS (
        SELECT 1
        FROM public."Submits" as s5
        JOIN public."Submission" as s6 ON s6."submissionID" = s5."submissionID"
        WHERE s6."verdict" = 'Accepted' AND s5."username" = u."contestantUsername" AND
s5."problemNumber" = 3
    ) as C,
    -- Subquery D
    EXISTS (
        SELECT 1
        FROM public."Submits" as s7
        JOIN public."Submission" as s8 ON s8."submissionID" = s7."submissionID"
        WHERE s8."verdict" = 'Accepted' AND s7."username" = u."contestantUsername" AND
s7."problemNumber" = 4
    ) as D,
    -- Subquery E
    EXISTS (
        SELECT 1
        FROM public."Submits" as s9
        JOIN public."Submission" as s10 ON s10."submissionID" = s9."submissionID"
        WHERE s10."verdict" = 'Accepted' AND s9."username" = u."contestantUsername" AND
s9."problemNumber" = 5
    ) as E,
    -- Subquery F
    EXISTS (
        SELECT 1
        FROM public."Submits" as s11
        JOIN public."Submission" as s12 ON s12."submissionID" = s11."submissionID"
```

```
      WHERE s12."verdict" = 'Accepted' AND s11."username" = u."contestantUsername" AND
s11."problemNumber" = 6
   ) as F
FROM
   (
      -- Subquery u
      SELECT
         c."contestantUsername",
         ci."score"
      FROM
         public."Contestant" as c
      JOIN
         public."CompetesIn" as ci ON c."contestantUsername" = ci."contestantUsername"
      WHERE
         ci."roundNumber" = '1'
   ) as u
ORDER BY
   u."score" DESC;
```

```sql
SELECT
    u."contestantUsername",
    -- Subquery A
    EXISTS (
        SELECT 1
        FROM public."Submits" as s1
        JOIN public."Submission" as s2 ON s2."submissionID" = s1."submissionID"
        WHERE s2."verdict" = 'Accepted' AND s1."username" = u."contestantUsername" AND s1."problemNumber" = 1
    ) as A,
    -- Subquery B
    EXISTS (
        SELECT 1
        FROM public."Submits" as s3
        JOIN public."Submission" as s4 ON s4."submissionID" = s3."submissionID"
        WHERE s4."verdict" = 'Accepted' AND s3."username" = u."contestantUsername" AND s3."problemNumber" = 2
    ) as B,
    -- Subquery C
    EXISTS (
        SELECT 1
        FROM public."Submits" as s5
        JOIN public."Submission" as s6 ON s6."submissionID" = s5."submissionID"
        WHERE s6."verdict" = 'Accepted' AND s5."username" = u."contestantUsername" AND s5."problemNumber" = 3
    ) as C,
    -- Subquery D
    EXISTS (
        SELECT 1
        FROM public."Submits" as s7
        JOIN public."Submission" as s8 ON s8."submissionID" = s7."submissionID"
        WHERE s8."verdict" = 'Accepted' AND s7."username" = u."contestantUsername" AND s7."problemNumber" = 4
    ) as D,
    -- Subquery E
    EXISTS (
        SELECT 1
        FROM public."Submits" as s9
        JOIN public."Submission" as s10 ON s10."submissionID" = s9."submissionID"
        WHERE s10."verdict" = 'Accepted' AND s9."username" = u."contestantUsername" AND s9."problemNumber" = 5
    ) as E,
    -- Subquery F
    EXISTS (
        SELECT 1
        FROM public."Submits" as s11
        JOIN public."Submission" as s12 ON s12."submissionID" = s11."submissionID"
        WHERE s12."verdict" = 'Accepted' AND s11."username" = u."contestantUsername" AND s11."problemNumber" = 6
    ) as F
FROM
    (
        -- Subquery u
        SELECT
            c."contestantUsername",
            ci."score"
        FROM
            public."Contestant" as c
        JOIN
            public."CompetesIn" as ci ON c."contestantUsername" = ci."contestantUsername"
        WHERE
            ci."roundNumber" = '1'
    ) as u
ORDER BY
    u."score" DESC;
```

contestantUsername [PK] text	a boolean	b boolean	c boolean	d boolean	e boolean	f boolean
Khaled47	true	true	false	false	false	false
ay_charaf	true	false	false	false	true	false
cutiepie	true	true	true	false	false	false
TheJoker	true	false	false	false	false	false

11.6.9- TagQuest

Admin MK aims to keep Dakowdas's user community well-informed about the problem topics that experience the least success in successful solutions. The intention is to encourage users to collaboratively focus on these challenging areas, fostering a collaborative effort to enhance proficiency in synonymous problem-solving.

```
INSERT INTO public."Announcement"(
   username, language, content, "timestamp")
VALUES (
   'MK',
   'English',
   CONCAT(
      CONCAT(
         (SELECT t."tag" as submissions
         FROM public."ProgrammingProblem" as p
         JOIN public."Tag" as t ON p."problemID" = t."problemID"
         JOIN public."Submits" as s ON s."problemNumber" = p."problemID"
         JOIN public."Submission" as s2 ON s2."submissionID" = s."submissionID"
         WHERE s2."verdict" = 'Accepted'
         GROUP BY t."tag"
         ORDER BY COUNT(t."tag") DESC
         LIMIT 1),
         ' was the tag least solved recently with '
      ),
      CAST(
         (SELECT COUNT(t."tag") as submissions
         FROM public."ProgrammingProblem" as p
         JOIN public."Tag" as t ON p."problemID" = t."problemID"
         JOIN public."Submits" as s ON s."problemNumber" = p."problemID"
         JOIN public."Submission" as s2 ON s2."submissionID" = s."submissionID"
```

```
        WHERE s2."verdict" = 'Accepted'
        GROUP BY t."tag"
        ORDER BY submissions
        LIMIT 1) AS VARCHAR
    )
  ),
  CURRENT_TIMESTAMP
);
```

```
INSERT INTO public."Announcement"(
    username, language, content, "timestamp")
VALUES (
    'MK',
    'English',
    CONCAT(
        CONCAT(
            (SELECT t."tag" as submissions
            FROM public."ProgrammingProblem" as p
            JOIN public."Tag" as t ON p."problemID" = t."problemID"
            JOIN public."Submits" as s ON s."problemNumber" = p."problemID"
            JOIN public."Submission" as s2 ON s2."submissionID" = s."submissionID"
            WHERE s2."verdict" = 'Accepted'
            GROUP BY t."tag"
            ORDER BY COUNT(t."tag") DESC
            LIMIT 1),
            ' was the tag least solved recently with '
        ),
        CAST(
            (SELECT COUNT(t."tag") as submissions
            FROM public."ProgrammingProblem" as p
            JOIN public."Tag" as t ON p."problemID" = t."problemID"
            JOIN public."Submits" as s ON s."problemNumber" = p."problemID"
            JOIN public."Submission" as s2 ON s2."submissionID" = s."submissionID"
            WHERE s2."verdict" = 'Accepted'
            GROUP BY t."tag"
            ORDER BY submissions
            LIMIT 1) AS VARCHAR
        )
    ),
     CURRENT_TIMESTAMP
 );
```

announcementID [PK] integer	username text	language text	content text	timestamp timestamp without time zone
11	MK	English	DP was the tag least solved recently with 1	2023-11-11 23:19:49.618007

11.6.10- Users In Battles

We aim to assess the performance of users within battle scenarios, gaining insights into their effectiveness and achievements during these engagements. This involves analyzing various metrics such as the number of problems successfully solved in battles, the total number of battles participated in, as well as distinguishing between victories and losses. This comprehensive evaluation provides a detailed perspective on users' contributions and success rates within the context of battles.

```
SELECT u."username",
  (
    SELECT COUNT(*)
    FROM public."Submits" s
    JOIN public."Submission" ss ON ss."submissionID" = s."submissionID"
    JOIN public."Joins" j ON j."username" = 'Danosour'
    JOIN public."ConsistsOf" c ON c."battleID" = j."battleID"
    join public."Battle" b on b."battleID" = c."battleID"
                WHERE s."username" = u."username" AND ss."verdict" = 'Accepted' AND
s."problemNumber" = c."problemID"
                and ss."timestamp" >= b."startTimestamp" and ss."timestamp" <=
b."endTimestamp"
  ) AS "numberOfProblemSolvedInBattles",

  (
    SELECT COUNT(j."battleID")
    FROM public."Joins" j
    WHERE j."username" = u."username"
  ) AS "TotalBattles",
```

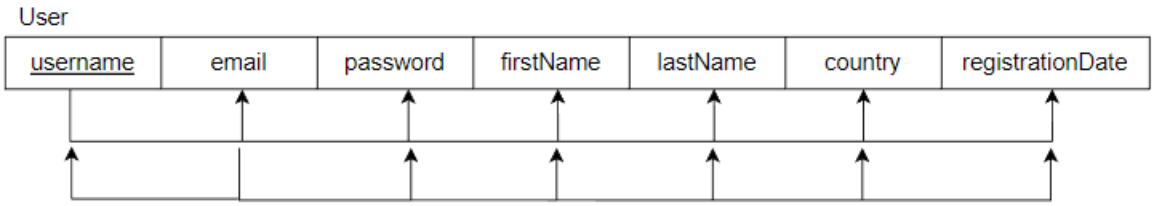
```
(
    SELECT SUM(CASE WHEN j."isWinner" = TRUE THEN 1 ELSE 0 END)
    FROM public."Joins" j
    WHERE j."username" = u."username"
) AS "Wins",

(
    SELECT SUM(CASE WHEN j."isWinner" = FALSE THEN 1 ELSE 0 END)
    FROM public."Joins" j
    WHERE j."username" = u."username"
) AS "Losses"
        from public."User" u;
```

```sql
SELECT u."username",
    (
        SELECT COUNT(*)
        FROM public."Submits" s
        JOIN public."Submission" ss ON ss."submissionID" = s."submissionID"
        JOIN public."Joins" j ON j."username" = 'Danosour'
        JOIN public."ConsistsOf" c ON c."battleID" = j."battleID"
        join public."Battle" b on b."battleID" = c."battleID"
        WHERE s."username" = u."username" AND ss."verdict" = 'Accepted' AND s."problemNumber" = c."problemID"
        and ss."timestamp" >= b."startTimestamp" and ss."timestamp" <= b."endTimestamp"
    ) AS "numberOfProblemSolvedInBattles",

    (
        SELECT COUNT(j."battleID")
        FROM public."Joins" j
        WHERE j."username" = u."username"
    ) AS "TotalBattles",

    (
        SELECT SUM(CASE WHEN j."isWinner" = TRUE THEN 1 ELSE 0 END)
        FROM public."Joins" j
        WHERE j."username" = u."username"
    ) AS "Wins",

    (
        SELECT SUM(CASE WHEN j."isWinner" = FALSE THEN 1 ELSE 0 END)
        FROM public."Joins" j
        WHERE j."username" = u."username"
    ) AS "Losses"
    from public."User" u;
```

username [PK] text	numberOfProblemSolvedInBattles bigint	TotalBattles bigint	Wins bigint	Losses bigint
Khaled47	0	2	1	1
AliFarhat	0	1	0	1
ay_charaf	0	1	1	0
HussKiller	0	2	0	2
ImanDestroyer	4	3	2	1
Danosour	4	3	1	2
HashTrash	0	4	2	2
TheJoker	0	1	0	1
MommyBoy	0	2	2	0
cutiepie	0	1	0	1
Tourist	0	2	2	0

12-Normalization

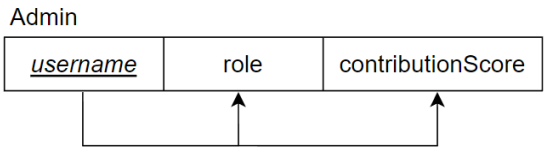
12.1- User



- **1NF:** User relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** User relation is in the second normal form (2NF) since it is in 1NF and all the non-prime attributes (all the attributes except for username and email) and email are fully functionally dependent on the primary key username.
- **3NF:** User relation is in the third normal form (3NF) since first it is in 2NF. The non-prime attributes (password, firstName, lastName, country, and registration date) are transitively dependent on the primary key username through the attributes email. Nevertheless, email is a candidate key (it's unique for every tuple in the User relation), hence there is no problem with the transitive dependency.

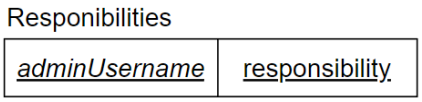
- **BCNF:** User relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and the determinants in ALL the functional dependencies are username, the primary key, and email, a candidate key.

12.2- Admin



- **1NF:** Admin relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Admin relation is in the second normal form (2NF) since it is in 1NF and all the non-prime attributes (role and contributionScore) are fully functionally dependent on the primary key username.
- **3NF:** User relation is in the third normal form (3NF) since first it is in 2NF and the non-prime attributes, role and contributionScore, are not transitively dependent on the primary key username.
- **BCNF:** User relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and the determinant in ALL the functional dependencies is username, the primary key.

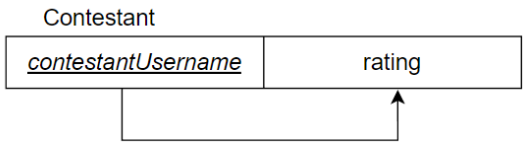
12.3- Responsibilities



- **1NF:** Responsibilities relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Responsibilities relation is in the second normal form (2NF) since it is in 1NF and there are no functional dependencies (no non-prime attributes).
- **3NF:** Responsibilities relation satisfies the third normal form (3NF) since first it is in 2NF and there are no functional dependencies (no non-prime attributes).

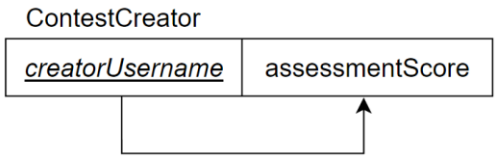
- **BCNF:** Responsibilities relation satisfies the Boyce-Codd normal form (BCNF) since it is in 3NF and there are no functional dependencies (no non-prime attributes).

12.4- Contestant



- **1NF:** Contestant relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Contestant relation is in the second normal form (2NF) since it is in 1NF and all the non-prime attributes (rating) are fully functionally dependent on the primary key contestantUsername.
- **3NF:** Contestant relation satisfies the third normal form (3NF) since it is in 2NF and the non-prime attribute rating is not transitively dependent on the primary key contestantUsername.
- **BCNF:** Contestant relation satisfies the Boyce-Codd normal form (BCNF) since it is in 3NF and the determinant in the functional dependency is contestantUsername which is the primary key.

12.5- ContestCreator



- **1NF:** ContestCreator relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** ContestCreator relation is in the second normal form (2NF) since it is in 1NF and the non-prime attribute, assesmentScore, is fully functionally dependent on the primary key creatorUsername.
- **3NF:** ContestCreator relation is in the third normal form (3NF) since it is in 2NF and the non-prime attribute assessmentScore is not transitively dependent on the primary key creatorUsername.
- **BCNF:** ContestCreator relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and the determinant in the functional dependency is creatorUsername, which is the primary key.

12.6- Organizes

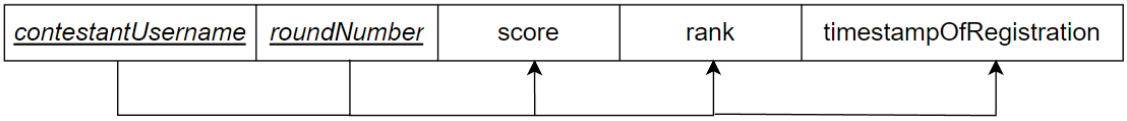
Organizes

creatorUsername	roundNumber

- **1NF:** Organizes relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Organizes relation is in the second normal form (2NF) since it is in 1NF and there are no functional dependencies (no non-prime attributes).
- **3NF:** Organizes relation is in the third normal form (3NF) since first it is in 2NF and there are no functional dependencies (no non-prime attributes).
- **BCNF:** Organizes relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and there are no functional dependencies (no non-prime attributes).

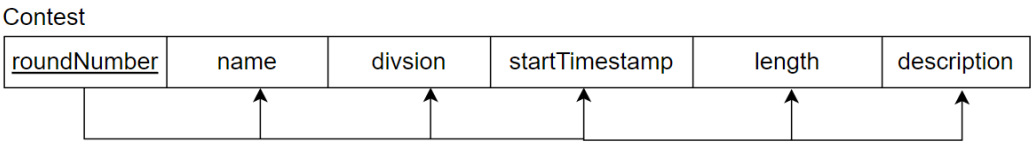
12.7- CompetesIn

CompetesIn



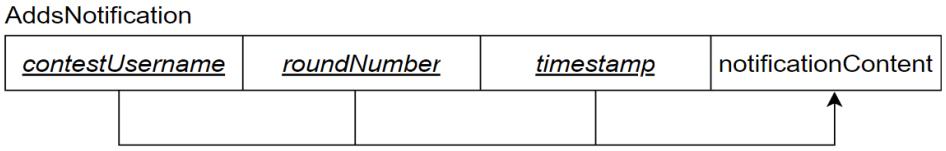
- **1NF:** CompetesIn relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** CompetesIn relation is in the second normal form (2NF) since it is in 1NF and the non-prime attributes (score, rank and timestampOfRegistration) are fully functionally dependent on the primary key which consists of the prime attributes contestantUsername & roundNumber.
- **3NF:** CompetesIn relation is in the third normal form (3NF) since it is in 2NF and the non-prime attributes (score, rank and timestampOfRegistration) are not transitively dependent on the primary key which consists of the prime attributes contestantUsername & roundNumber.
- **BCNF:** CompetesIn relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and the determinants in ALL the functional dependencies are the prime attributes contestantUsername & roundNumber, which constitute the primary key.

12.8- Contest



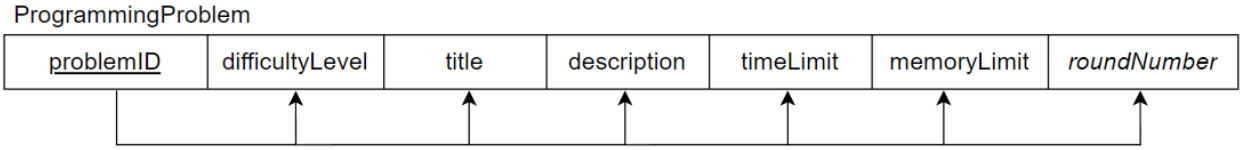
- **1NF:** Contest relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Contest relation is in the second normal form (2NF) since it is in 1NF and all the non-prime attributes (name, division, startTimestamp, length and description) are fully functionally dependent on the primary key roundNumber.
- **3NF:** Contest relation is in the third normal form (3NF) since first it is in 2NF and the non-prime attributes (name, division, startTimestamp, length and description) are not transitively dependent on the primary key roundNumber.
- **BCNF:** Contest relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and the determinant in ALL the functional dependencies is roundNumber, the primary key.
-

12.9- AddsNotification



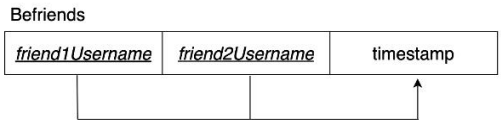
- **1NF:** AddsNotification relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** AddsNotification relation is in the second normal form (2NF) since it is in 1NF and the non-prime attribute notificationContent is fully functionally dependent on the primary key which consists of the prime attributes contestantUsername, roundNumber, & timestamp.
- **3NF:** AddsNotification relation is in the third normal form (3NF) since it is in 2NF and the non-prime attribute notificationContent is not transitively dependent on the primary key which consists of the prime attributes contestantUsername, roundNumber, & timestamp.
- **BCNF:** AddsNotification relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and the determinants in the functional dependency are the prime attributes contestantUsername, roundNumber, & timestamp, which constitute the primary key.

12.10- ProgrammingProblem



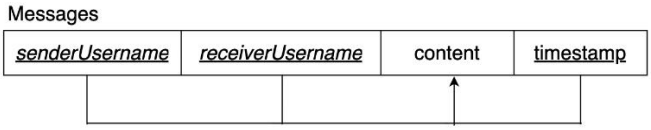
- **1NF:** ProgrammingProblem relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** ProgrammingProblem relation is in the second normal form (2NF) since it is in 1NF and all the non-prime attributes (dificultyLevel, title, description, timeLimit, memoryLimit and roundNumber) are fully functionally dependent on the primary key problemID.
- **3NF:** ProgrammingProblem relation is in the third normal form (3NF) since first it is in 2NF and the non-prime attributes (dificultyLevel, title, description, timeLimit, memoryLimit and roundNumber) are not transitively dependent on the primary key problemID.
- **BCNF:** ProgrammingProblem relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and the determinant in ALL the functional dependencies is problemID, the primary key.

12.11- Befriends



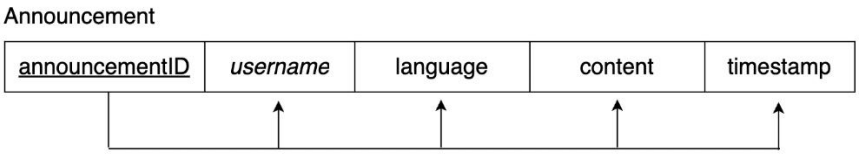
- **1NF:** Befriends relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Befriends relation is in the second normal form (2NF) since it is in 1NF and the non-prime attribute (timestamp) is fully functionally dependent on the primary key which consists of the prime attributes frined1Username & friend2Username.
- **3NF:** Befriends relation is in the third normal form (3NF) since it is in 2NF and the non-prime attribute (timestamp) is not transitively dependent on the primary key which consists of the prime attributes frined1Username & friend2Username.
- **BCNF:** Befriends relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and the determinants in ALL the functional dependencies are the prime attributes frined1Username & friend2Username, which constitute the primary key.

12.12- Messages



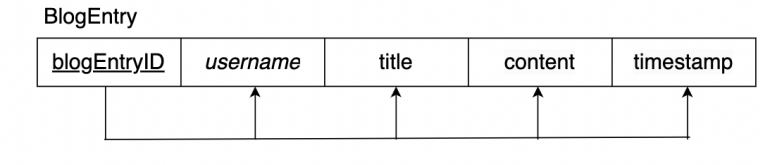
- **1NF:** Messages relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Messages relation is in the second normal form (2NF) since it is in 1NF and the non-prime attribute (content) is fully functionally dependent on the primary key which consists of the prime attributes senderUsername, receiverUsername, & timestamp.
- **3NF:** Messages relation is in the third normal form (3NF) since it is in 2NF and the non-prime attribute (content) is not transitively dependent on the primary key which consists of the prime attributes senderUsername, receiverUsername, & timestamp.
- **BCNF:** Messages relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and the determinants in ALL the functional dependencies are the prime attributes senderUsername, receiverUsername, & timestamp, which constitute the primary key.

12.13- Announcement



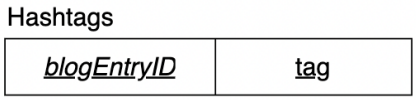
- **1NF:** Announcement relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Announcement relation is in the second normal form (2NF) since it is in 1NF and the non-prime attributes (username, language, content, and timestamp) are fully functionally dependent on the primary key which consists of the prime attribute announcementID.
- **3NF:** Announcement relation is in the third normal form (3NF) since it is in 2NF and the non-prime attributes (username, language, content, and timestamp) are not transitively dependent on the primary key which consists of the prime attribute announcementID.
- **BCNF:** Announcement relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and the determinants in ALL the functional dependencies is the prime attribute announcementID, the primary key.

12.14- BlogEntry



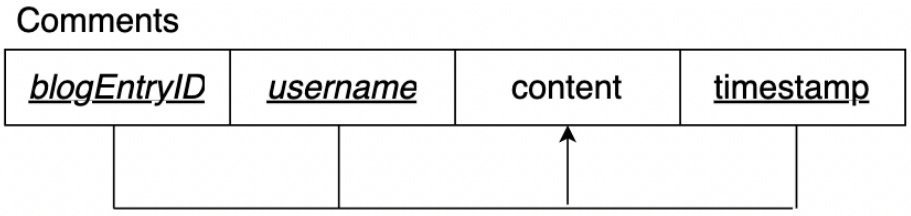
- **1NF:** BlogEntry relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** BlogEntry relation is in the second normal form (2NF) since it is in 1NF and the non-prime attributes (username, title, content and timestamp) are fully functionally dependent on the primary key which consists of the prime attribute blogEntryID.
- **3NF:** BlogEntry relation is in the third normal form (3NF) since it is in 2NF and the non-prime attributes (username, title, content and timestamp) are not transitively dependent on the primary key which consists of the prime attribute blogEntryID**.**
- **BCNF:** BlogEntry relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and the determinants in ALL the functional dependencies is the prime attribute blogEntryID, the primary key.

12.15- Hashtags



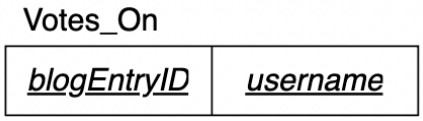
- **1NF:** Hashtags relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Hashtags relation is in the second normal form (2NF) since it is in 1NF and there are no functional dependencies (no non-prime attributes).
- **3NF:** Hashtags relation is in the third normal form (3NF) since first it is in 2NF and there are no functional dependencies (no non-prime attributes).
- **BCNF:** Hashtags relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and there are no functional dependencies (no non-prime attributes).

12.16- Comments



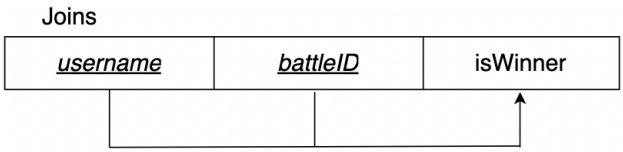
- **1NF:** Comments relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Comments relation is in the second normal form (2NF) since it is in 1NF and the non-prime attribute (content) is fully functionally dependent on the primary key which consists of the prime attributes blogEntryID, username, & timestamp.
- **3NF:** Comments relation is in the third normal form (3NF) since it is in 2NF and the non-prime attribute (content) is not transitively dependent on the primary key which consists of the prime attributes blogEntryID, username, & timestamp.
- **BCNF:** Comments relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and the determinants in ALL the functional dependencies are the prime attributes blogEntryID, username, & timestamp, which constitute the primary key.

12.17- Votes_On



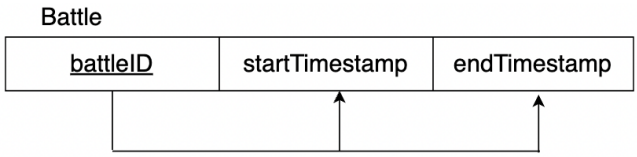
- **1NF:** Votes_On relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Votes_On relation is in the second normal form (2NF) since it is in 1NF and there are no functional dependencies (no non-prime attributes).
- **3NF:** Votes_On relation is in the third normal form (3NF) since first it is in 2NF and there are no functional dependencies (no non-prime attributes).
- **BCNF:** Votes_On relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and there are no functional dependencies (no non-prime attributes).

12.18- Joins



- **1NF:** Joins relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Joins relation is in the second normal form (2NF) since it is in 1NF and the non-prime attribute (isWinner) is fully functionally dependent on the primary key which consists of the prime attributes username & battleID.
- **3NF:** Joins relation is in the third normal form (3NF) since it is in 2NF and the non-prime attribute (isWinner) is not transitively dependent on the primary key which consists of the prime attributes username & battleID.
- **BCNF:** Joins relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and the determinants in ALL the functional dependencies are the prime attributes username & battleID, which constitute the primary key.

12.19- Battle



- **1NF:** Battle relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Battle relation is in the second normal form (2NF) since it is in 1NF and the non-prime attributes (startTimestamp, and endTimestamp) are fully functionally dependent on the primary key which consists of the prime attribute battleID.
- **3NF:** Battle relation is in the third normal form (3NF) since it is in 2NF and the non-prime attributes (startTimestamp, and endTimestamp) are not transitively dependent on the primary key which consists of the prime attribute battleID**.**
- **BCNF:** Battle relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and the determinants in ALL the functional dependencies is the prime attribute battleID, the primary key.

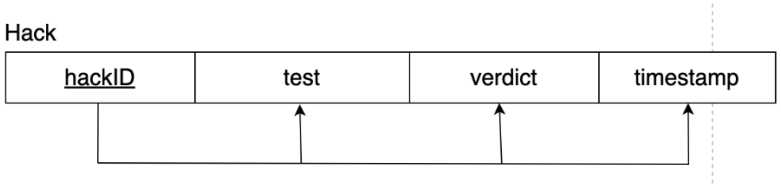
12.20- ConsistsOf

ConsistsOf

battleID	problemID

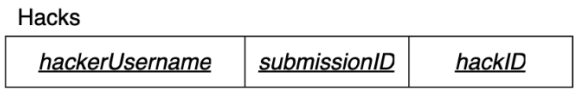
- **1NF:** ConsistsOf relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** ConsistsOf relation is in the second normal form (2NF) since it is in 1NF and there are no non-prime attributes.
- **3NF:** ConsistsOf relation is in the third normal form (3NF) since it is in 2NF and there are no non-prime attributes.
- **BCNF:** Befriends relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and there are no none-prime attributes.

12.21- Hack



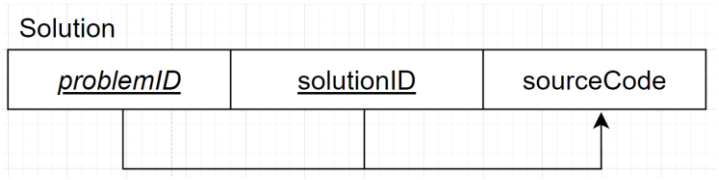
- **1NF:** Hack relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Hack relation is in the second normal form (2NF) since it is in 1NF and the non-prime attributes (test, verdict, timestamp) are fully functionally dependent on the primary key which consists of the prime attribute hackID.
- **3NF:** Hack relation is in the third normal form (3NF) since it is in 2NF and the non-prime attributes (test, verdict, timestamp) are not transitively dependent on the primary key which consists of the prime attribute hackID.
- **BCNF:** Hack relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and the determinants in ALL the functional dependencies is the prime attribute hackID which constitutes the primary key.

12.22- Hacks



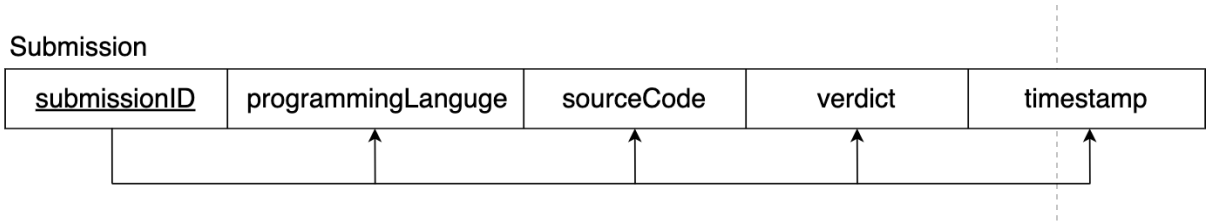
- **1NF:** Hacks relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Hacks relation is in the second normal form (2NF) since it is in 1NF and there are no non-prime attributes.
- **3NF:** Hacks relation is in the third normal form (3NF) since it is in 2NF and there are no non-prime attributes.
- **BCNF:** Hacks relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and there are no non-prime attributes.

12.23- Solution



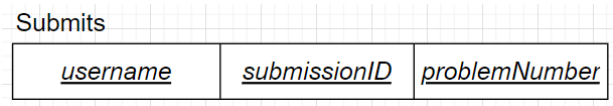
- **1NF:** Solution relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Solution relation is in the second normal form (2NF) since it is in 1NF and the non-prime attribute (sourceCode) are fully functionally dependent on the primary key which consists of the prime attributes problemID and solutionID .
- **3NF:** Solution relation is in the third normal form (3NF) since it is in 2NF and the non-prime attribute (sourceCode) are not transitively dependent on the primary key which consists of the prime attributes problemID and solutionID.
- **BCNF:** Solution relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and the determinants in ALL the functional dependencies are the prime attributes problemID and solutionID, which constitute the primary key.

12.24- Submission



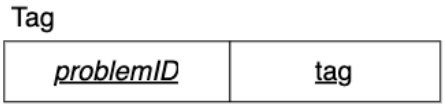
- **1NF:** Submission relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Submission relation is in the second normal form (2NF) since it is in 1NF and the non-prime attributes (programmingLanguge, sourceCode, verdict, timestamp) are fully functionally dependent on the primary key which consists of the prime attribute submissionID.
- **3NF:** Submission relation is in the third normal form (3NF) since first it is in 2NF and the non-prime attributes (programmingLanguge, sourceCode, verdict, timestamp) are not transitively dependent on the primary key which consists of the prime attribute submissionID.
- **BCNF:** Submission relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and the determinants in ALL the functional dependencies is the prime attribute submissionID which constitutes the primary key.

12.25- Submits



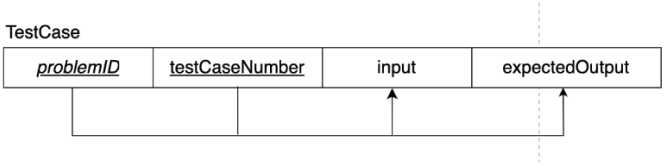
- **1NF:** Submits relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Submits relation is in the second normal form (2NF) since it is in 1NF and there are no non-prime attributes and no functional dependencies.
- **3NF:** Submits relation is in the third normal form (3NF) since it is in 2NF and there are no functional dependencies and no non-prime attributes hence none of the non-prime attributes are transitively dependent on the primary key.
- **BCNF:** Submits relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and there are no functional dependencies and no non-prime attributes

12.26- Tag



- **1NF:** Tag relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Tag relation is in the second normal form (2NF) since it is in 1NF and there are no functional dependencies (no non-prime attributes).
- **3NF:** Tag relation is in the third normal form (3NF) since first it is in 2NF and there are no functional dependencies (no non-prime attributes).
- **BCNF:** Tag relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and there are no functional dependencies (no non-prime attributes).

12.27- Testcase



- **1NF:** Testcase relation is in the first normal form (1NF) since the attribute values are single atomic (no composite/multivalued attributes). This is due to the utilization of the ER model.
- **2NF:** Testcase relation is in the second normal form (2NF) since it is in 1NF and the non-prime attributes (input,expectedOutput) are fully functionally dependent on the primary key which consists of the prime attributes (problemID and testcaseNumber)
- **3NF:** Testcase relation is in the third normal form (3NF) since it is in 2NF and the non-prime attributes (input,expectedOutput) are not transitively dependent on the primary key which consists of the prime attributes (problemID and testcaseNumber).
- **BCNF:** Testcase relation is in the Boyce-Codd normal form (BCNF) since it is in 3NF and the determinants in ALL the functional dependencies are the prime attributes problemID & testcaseNumber, which constitute the primary key.

13-Conclusion

After conducting a comprehensive analysis of our database structure for Dakowdas.com, we take pride in presenting our initial Entity-Relationship (ER) draft as well as its mapping and its implementation on pgAdmin. While we acknowledge the need for some future refinements and adjustments, we are pleased with the organized and coherent framework that our first draft represents. This draft lays a strong foundation for our coding website's database, reflecting our commitment to creating a robust and scalable platform. Our mission is to scale this website to become a front-tier competitive programming platform, and this initial draft is a significant stride towards achieving that goal. As we progress in our development journey, we will continue to refine and optimize the database to ensure it aligns seamlessly with our evolving needs and objectives, bolstering our vision to establish Dakowdas.com as a leading force in the competitive programming arena.

14-Instructor’s Feedback