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 1 of the database project for the course EECE433 – Database Systems

September 2023

Table of Contents

1 – Introduction
2 – References/Copyright
3 – Tool(s) used to draw the ER
4 – System Description & Requirements
5 - Legend of ER diagram symbols
6 – Complete ER Diagram for the Joy Zoological Park database
6.1 Entity Types & Their Attributes
6.2 Relationships and their explanations
7 – Conclusion
8 – Instructor Feedback

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 **Dakodas** 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, section 6 encompasses the entity types and relationships in detail. 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

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. A user has a rating which reflects his competence in problem solving. Based on this rating, each user is assigned a bracket. 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 follow another user on a certain following date.
- 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.

- 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 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 date, defender(user who submitted the submission), verdict (Invalid Input, Unsuccessful Hacking Attempt, or Successful

Hacking Attempt), 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

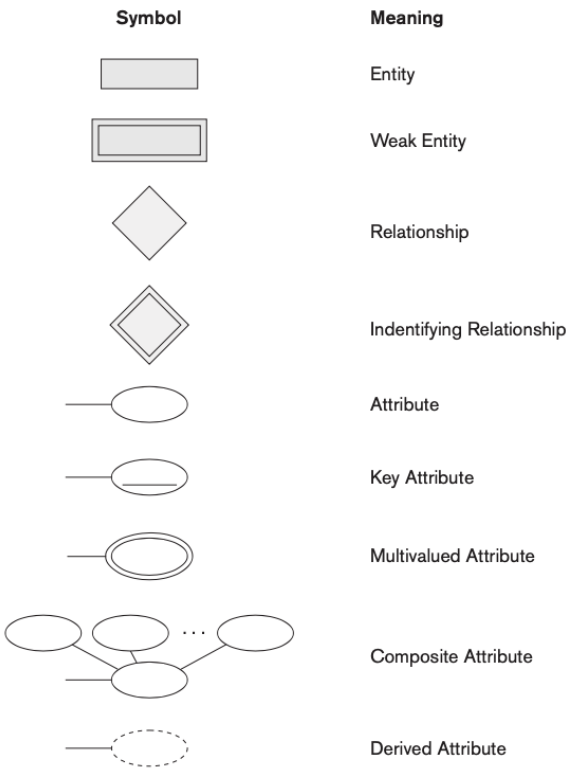
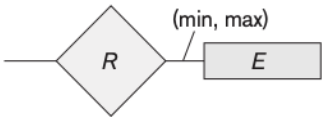


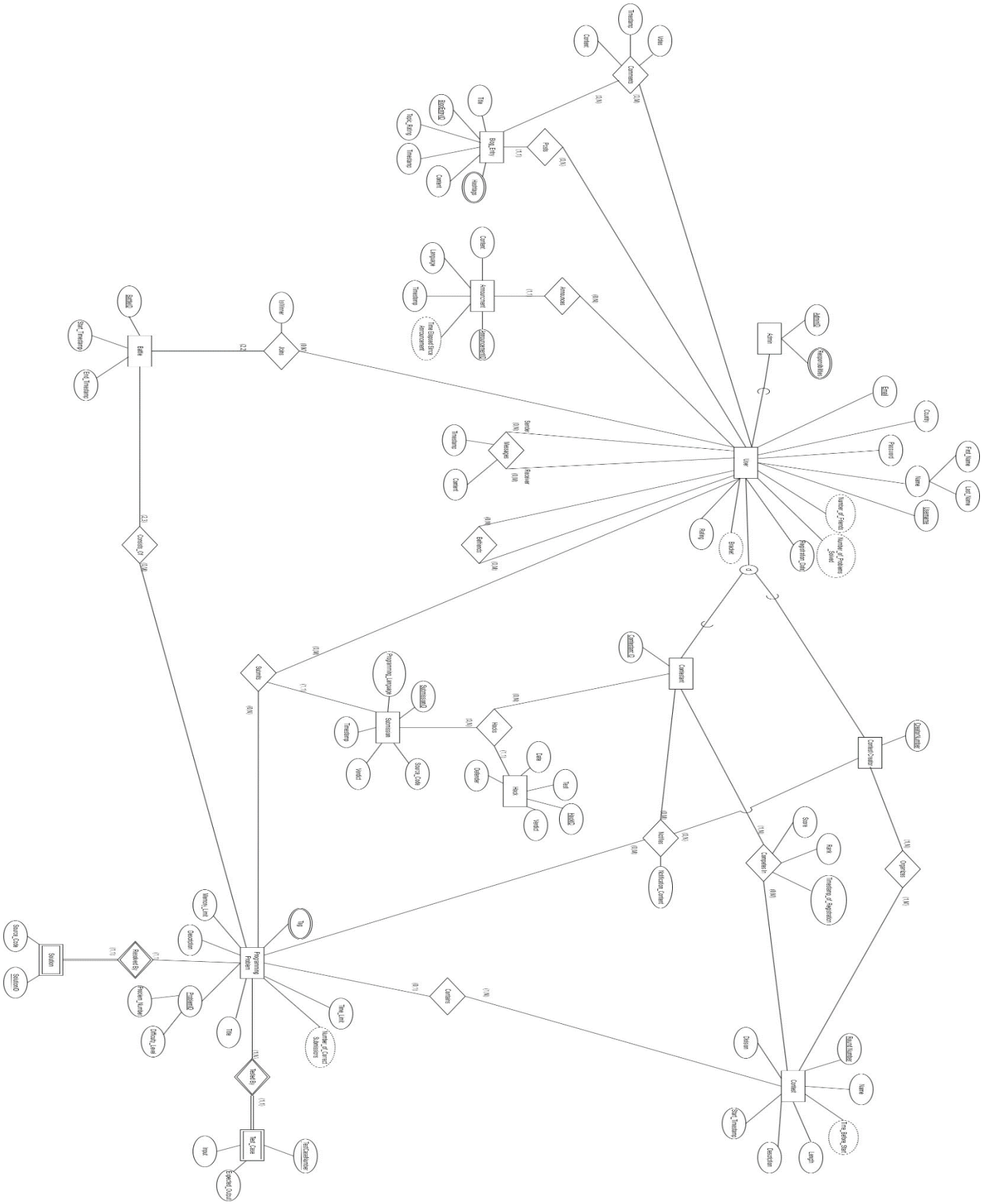
Figure 3.14 Summary of the notation for ER diagrams.



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

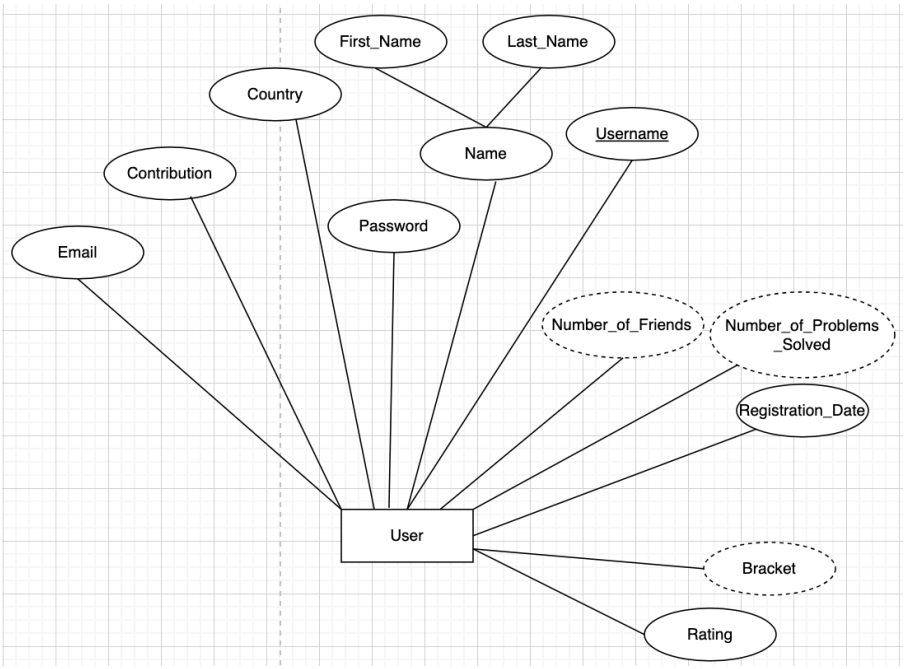
6 – Complete ER Diagram for the Joy Zoological Park database

URL:



6.1- Entity Types & Their Attributes

6.1.1- 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)Contribution: reflects how active the user is when it comes to interacting with other users on the platform (via comments or relevant blog entries)

6)Registration Date

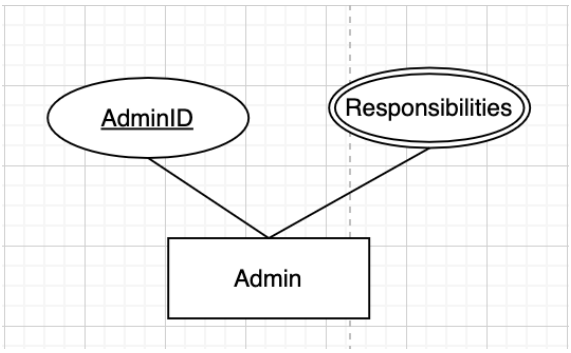
7)Rating: demonstrates the prowess of the user.

8)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 user.

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

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

6.1.2- Admin:



-Entity Name: Admin

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

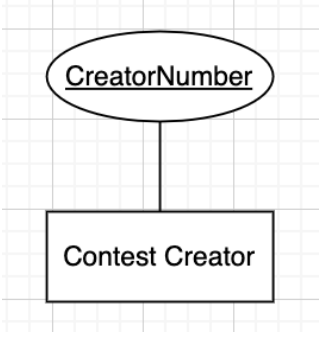
-Entity Type: strong

-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.

6.1.3- Contest Creator:



-Entity Name: Contest Creator

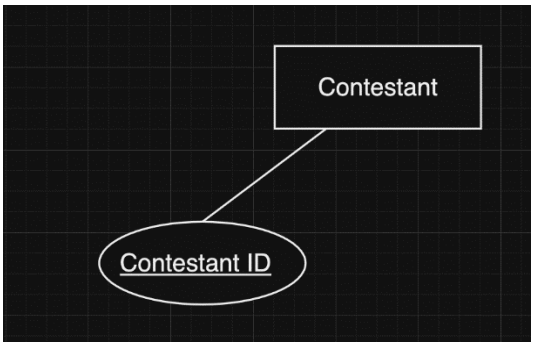
Contests are organized by contest creators. This distinguishes a normal user from a contest creator.

-Entity Type: strong

-Key Attribute(s): Creator Number

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

6.1.4- Contestant:



-Entity Name: Contestant

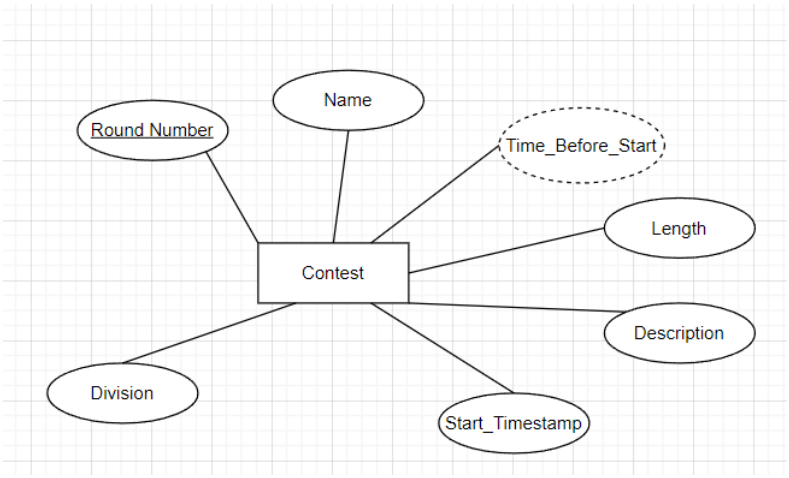
This entity represents those who compete in contests.

-Entity Type: strong

-Key Attribute(s): Contestant ID

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

6.1.5- 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.

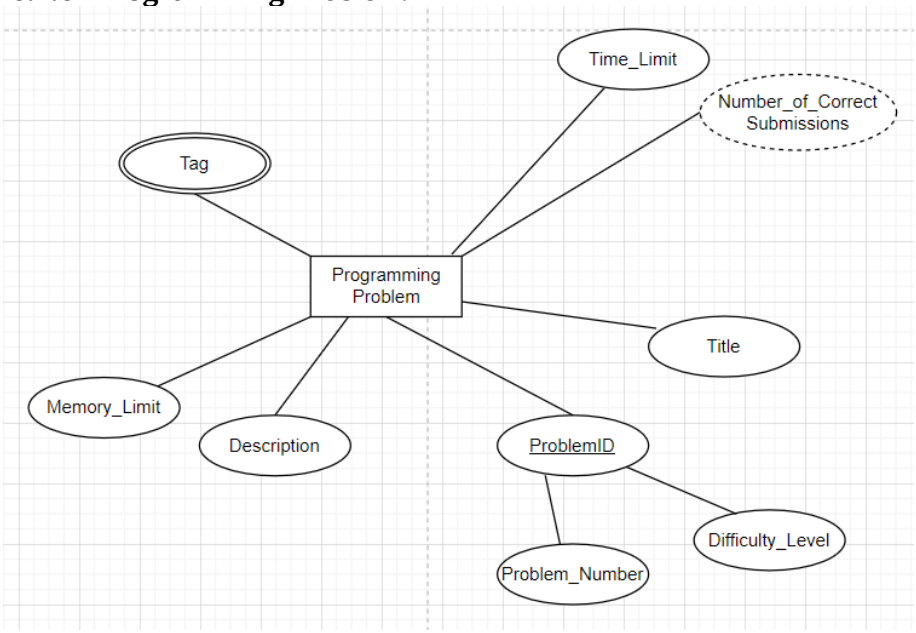
3)Start Timestamp: the date and time when the contest will begin. This is typically provided in Coordinated Universal Time (UTC).

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.

6.1.6- Programming Problem:



-Entity Name: Problem

-Entity Type: strong

-Key Attribute(s):Problem ID

Problem ID is a ***composite*** entity. It is composed of

>Problem Number: index of the problem with respect to other problems with the same difficulty level.

>Difficulty Level: ex:A, B, C, D,E,F.

The combination of the problem number and the difficulty level uniquely identifies the problem.

-Other Attributes:

1)Title

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

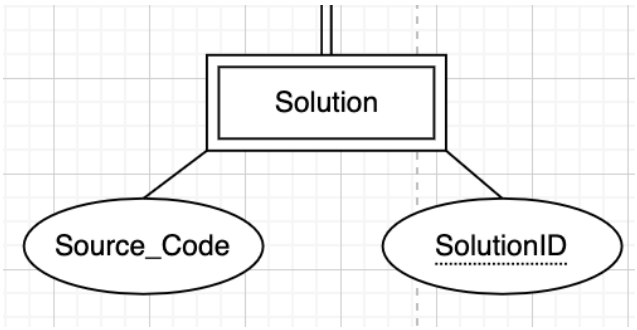
3)Memory Limit: maximum memory that a submission may use. If a submission exceeds the memory limit, then it is rejected. It's usually given in gigabytes.

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.

6.1.7- Solution:



-Entity Name: Solution

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

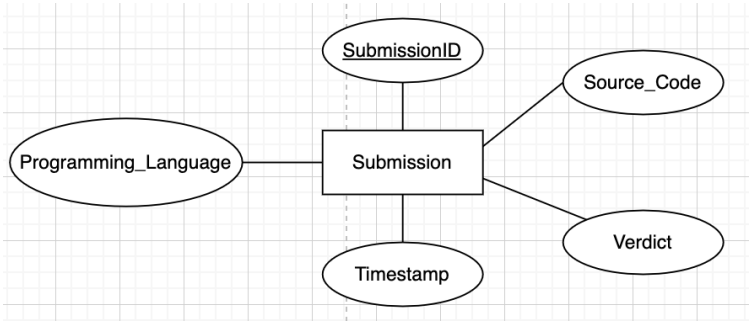
-Partial Key: Solution ID.

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.

6.1.8- 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.

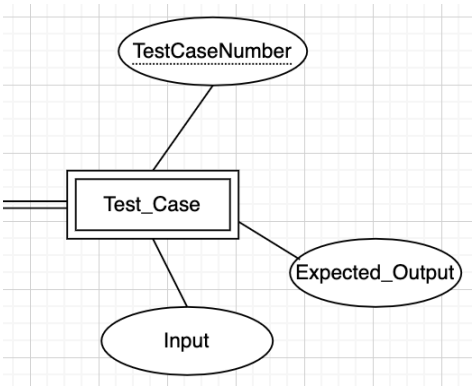
Verdicts are: -Accepted:source code ran correctly on all test cases.

-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.

6.1.9- Test Case:



-Entity Name: Test Case

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

-Partial Key: TestCaseNumber.

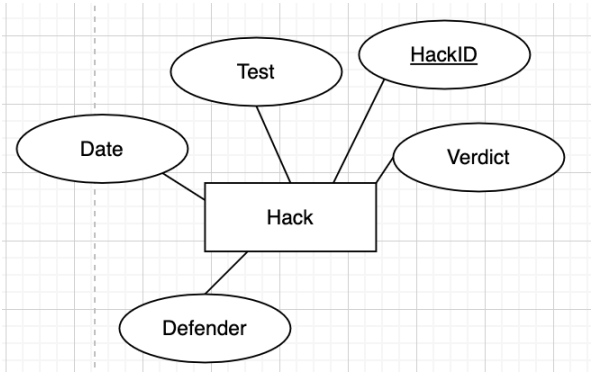
A solution is uniquely identified by the combination of the TestCaseNumeber and the identifying programming problem's ID.

-Other Attributes:

1)Input: input to the code

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

6.1.10- 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 penalised.

-Entity Type: strong

-Key Attribute(s): Hack ID

-Other Attributes:

1)Defender: contestant who submitted the submission in question.

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

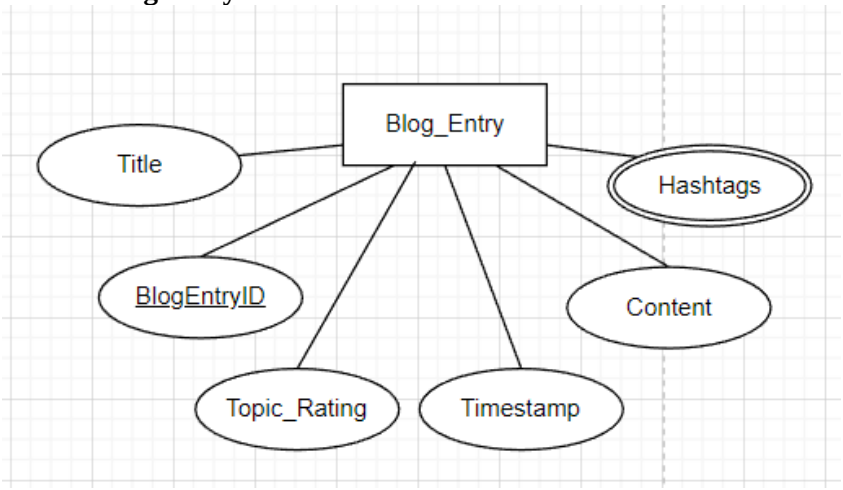
3)Verdict: -Invalid Input: input provided by the hacker doesn't meet the problem's input requirements.

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

-Successful Hacking Attempt submission failed on test.

4)Date: date on which the hacker submitted the hack.

6.1.11- 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:

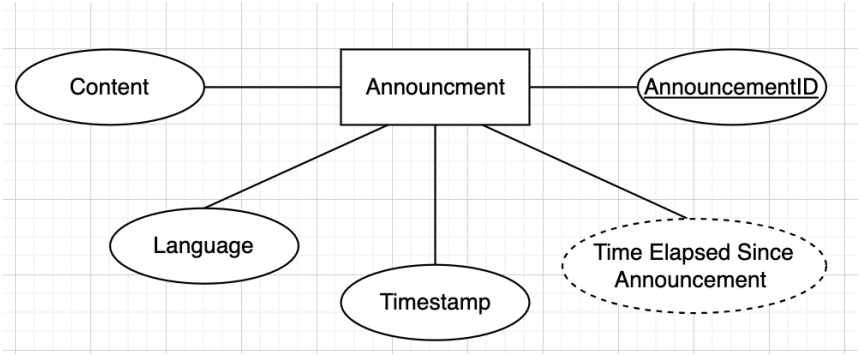
1)Title: reflects the topic discussed in the entry.

2)Content: actual text of the blog entry

3)Topic Rating: vote given by other users to assess the relevance of the topic discussed.

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.

6.1.12- Announcement:



-Entity Name: Announcement

Announcements are made by users to advertise contests.

-Entity Type: strong

-Key Attribute(s):AnnouncementID

-Other Attributes:

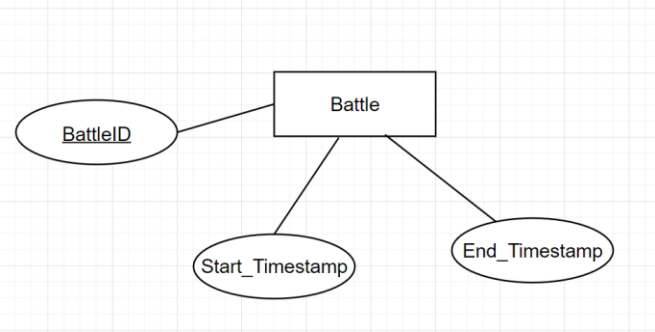
1)Content: actual text of the announcement

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.

6.1.13- 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

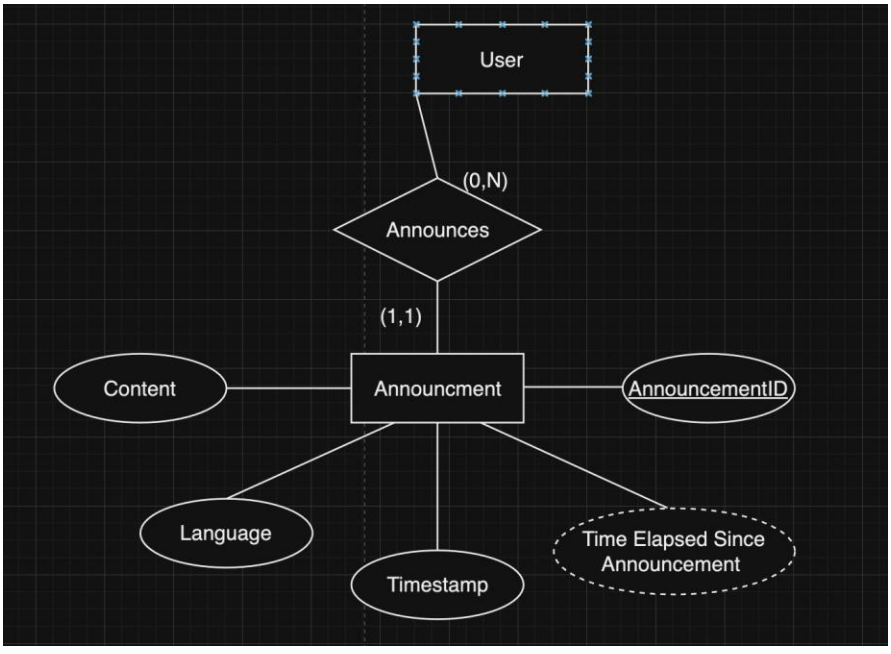
-Other Attributes:

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.

6.2- Relationships and Their Explanations

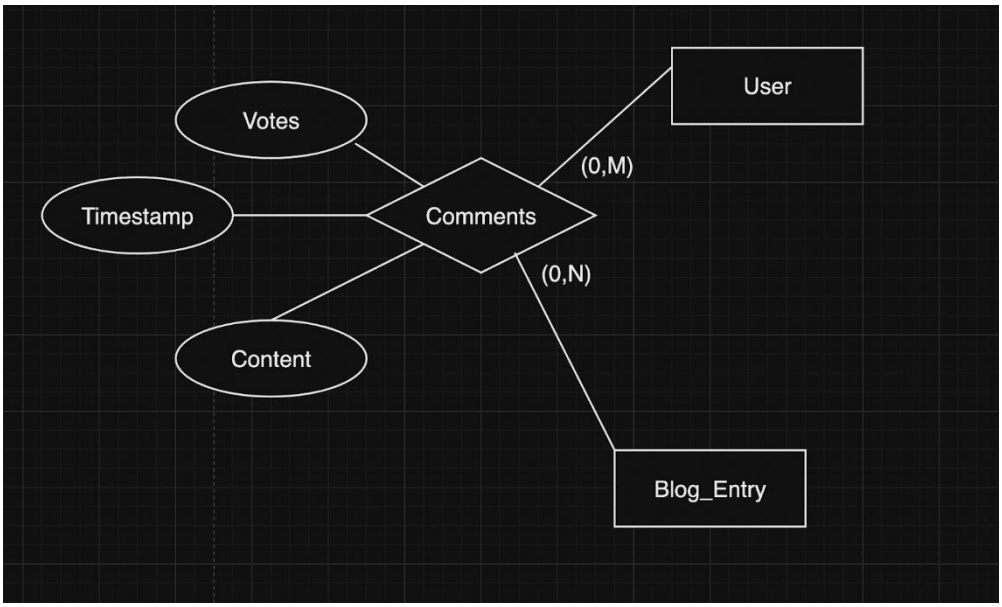
6.2.1- Announces



Relationship Type: Binary

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

6.2.2- Comments



Relationship Type: Binary

A user can comment in a Blog Entry.

A Blog Entry can have many comments.

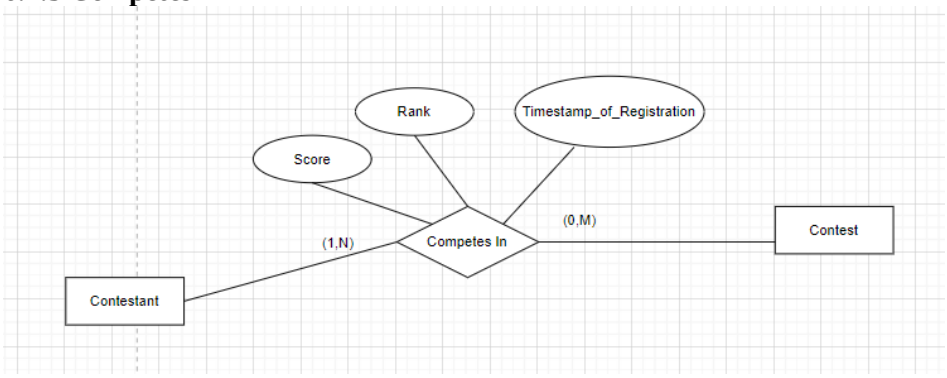
For each comment, we store:

-Content: information shared in the comment.

-Timestamp: time at which the comment was made

-Votes: other users' rating of the comment, which reflects their approval or dissatisfaction with the comment.

6.2.3 Competes

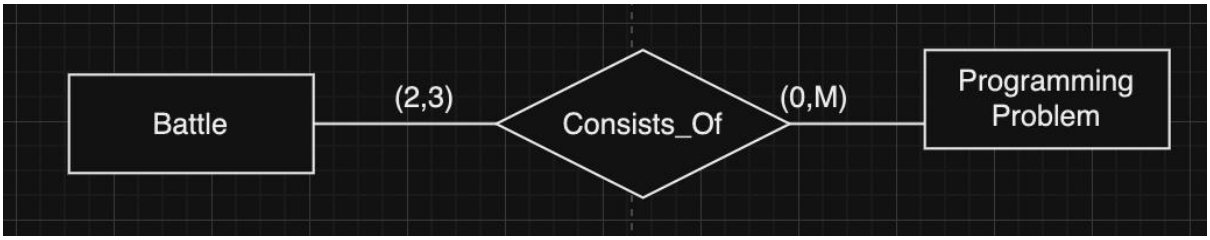


Relationship Type :Binary

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.

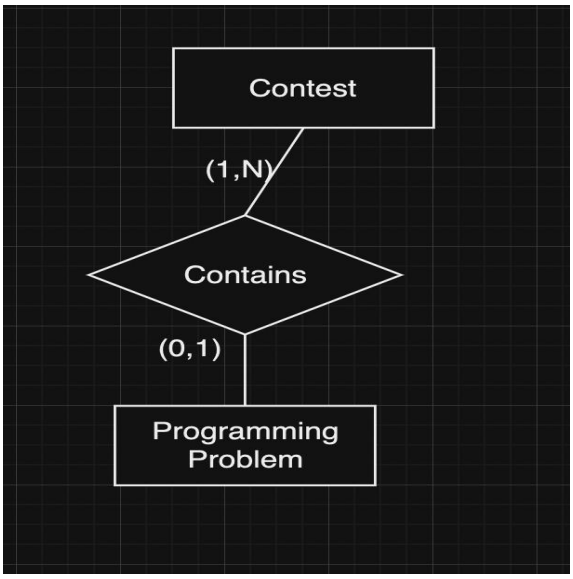
6.2.4 Consists Of



Relationship Type: Binary

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

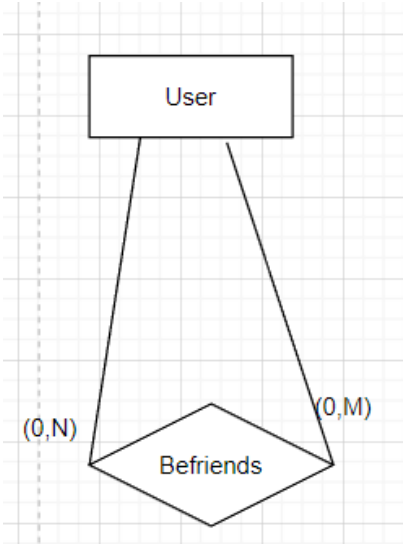
6.2.5- Consists Of



Relationship Type: Binary

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

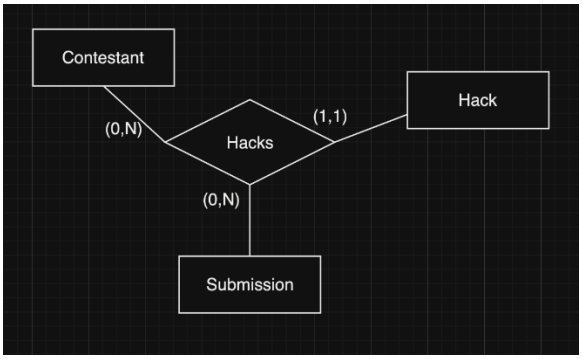
6.2.6- Befriends



Relationship Type: Recursive

A user can befriend many users.

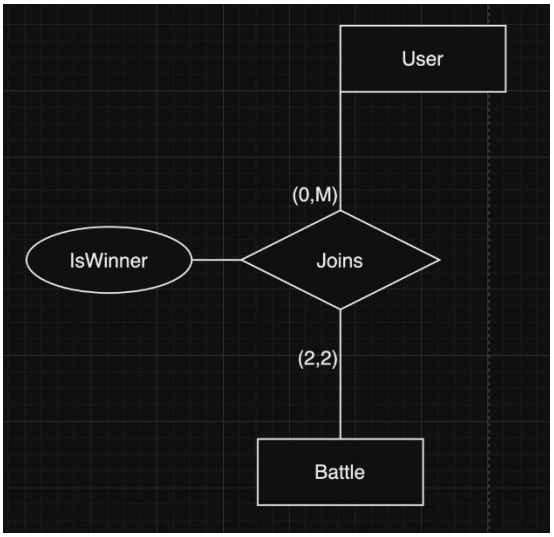
6.2.7- Hacks:



Relationship Type: Ternary

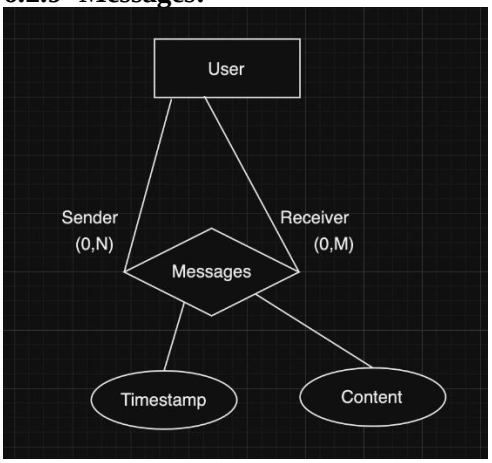
A contestant can hack many submissions with a hack. Any failed hacks are stored to penalise the hacker for an unsuccessful submission. Otherwise, the hacker is rewarded.

6.2.8- Joins:



Relationship Type: Binary
A user can join many battles. A battle consists of only 2 users.

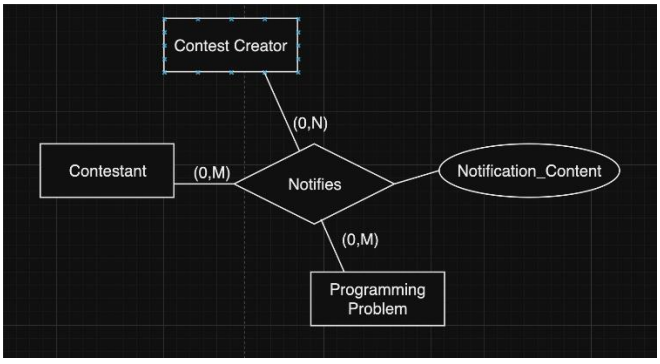
6.2.9- Messages:



Relationship Type: Recursive

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

6.2.10- Notifies:

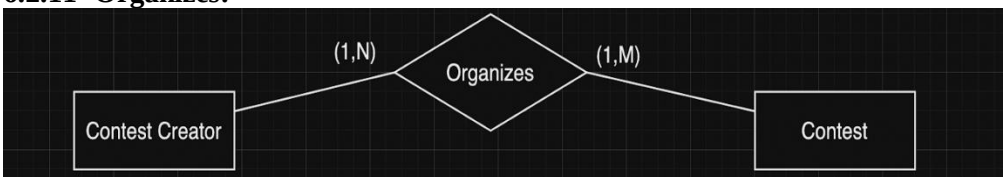


Relationship Type: Ternary

A contestant creator(s) can notify all contestants about any missing details for a programming problem.

The content of the notification is recorded.

6.2.11- Organizes:

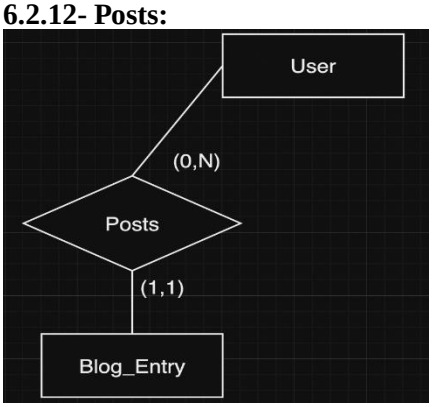


Relationship Type: Binary

A contest creator organizes at least one contest.

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

6.2.12- Posts:

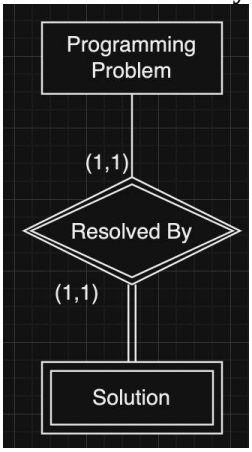


Relationship Type: Binary

A user can post many blog entries.

A blog entry must be posted by only one user.

6.2.13- Resolved By:

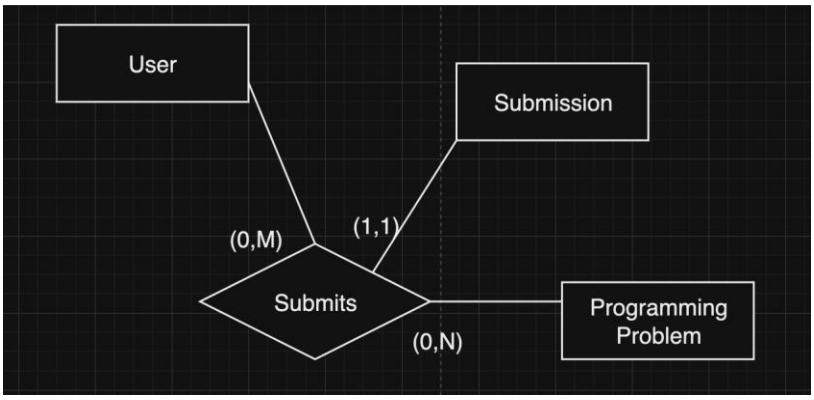


Relationship Type: Binary - Identifying Relationship

A programming problem is resolved by exactly one solution.

A solution resolves exactly one programming problem.

6.2.14- Submits:

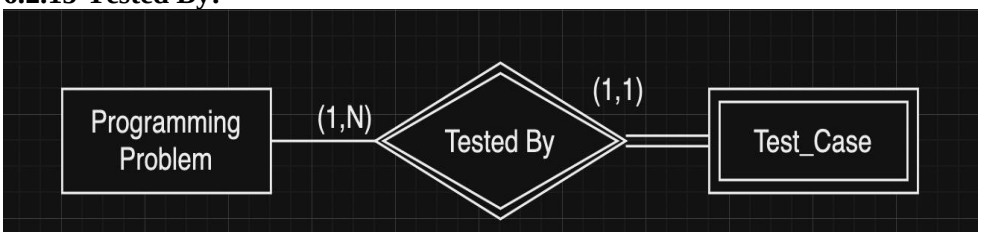


Relationship Type: Ternary

A user submits many submissions for many programming problems.

A submission is submitted by exactly one user.

6.2.15-Tested By:

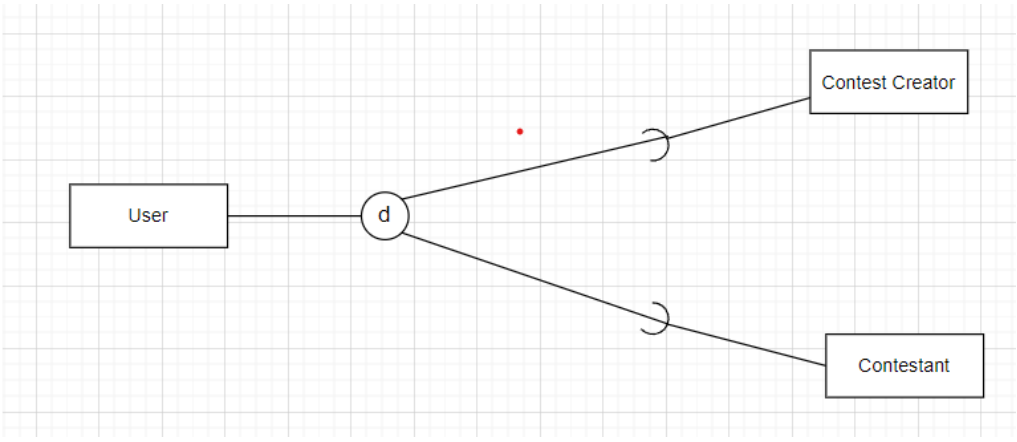


Relationship Type: Binary - Identifying Relationship

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

A test case tests only one programming problem.

6.2.16-Inheritance 1:

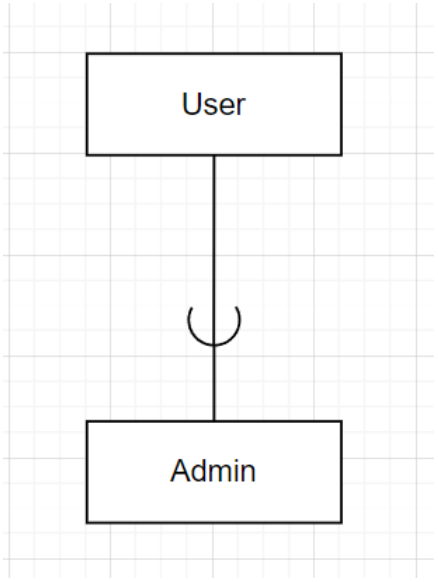


Parent: User

Children: Contestant & Contest Creator

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

6.2.17-Inheritance 2:



Parent: User

Children: Admin

An admin is a user with extra authority and responsibilities.

7-Conclusion

After conducting a comprehensive analysis of our database structure for Dakowdas.com, we take pride in presenting our initial Entity-Relationship (ER) draft. 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 ER 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.

8- Instructor Feedback