UNIVERSITATEA DIN BUCURESTI
FACULTATEA DE MATEMATICA SI INFORMATICA
CALCULATOARE SI TEHNOLOGIA INFORMATIEI

PROIECT BAZE DE DATE

PROFESOR COORDONATOR:
VASILE SILVIU-LAURENTIU

STUDENT:
LICU MIHAI-GEORGE

BUCURESTI 2022

Baza de date a unei platforme de concursuri CTF

- CTFLime -

PROFESOR COORDONATOR:
VASILE SILVIU-LAURENTIU

STUDENT:
LICU MIHAI-GEORGE

BUCURESTI 2022

CUPRINS

Prezentarea modelului (din lumea reala) si a regulilor acestuia

Modelul din lumea reala si motivatia alegerii acestuia

Tema aleasa de mine pentru proiectul la disciplina Baze de date este "Baza de date a unei platforme de concursuri CTF", numind aceasta platforma "CTFLime", o scriere gresita intentionata a numelui platformei originale de la care am preluat inspiratie "CTFTime".

CTF este un acronim pentru termenul Capture The Flag, aceste concursuri sunt competitii de cibersecuritate constituite din mai multe probleme din diverse domenii ale informaticii ce au ca scop exploatarea unei vulnerabilitati pentru a accesa informatii fara a avea permisiune directa, in aceste exercitii aceasta informatie este un asa numit "flag", un string de text de obicei incapsulat cu acolade si avand un prefix, posibil specific concursului. ex: CTF{some_text_here}

Am ales tema aceasta deoarece de un an am inceput sa particip la astfel de competitii pentru a imi largi domeniul cunostintelor legate de programare si pentru a dobandi abilitati ale unui ingier bun. In acest timp am reusit sa invat singur diverse concepte si am obtinut rezultate bune la competitii nationale si internationale.

Aceasta platforma pe care incerc sa o emulez ar fi, in primul rand, ca un "hub" unde persoanele ar veni sa vada competitiile din trecut cu problemele si rezolvarile acestora dar si competitiile viitoare care inca nu au luat loc, de asemenea topuri de scoruri pe tari sau diversi autori din domeniu si detalii despre acestia. In al doilea rand platforma ar gazdui si competitii actuale si ar asigura toate sistemele necesare astfel incat competitiile sa decurca bine, precum tracking de scor si incercarile echipelor participante. Astfel, am construit baza de date cu aceste nevoi in minte.

Regulile modelului

Fiecare utilizator nou initial nu va face parte din nici o echipa, dar are optiunea de a se alatura maxim unei singure echipe folosind codul secret generat de catre platforma acesteia. Un utilizator fara echipa nu poate participa la concursuri.

La crearea unei echipe aceasta este intial goala si nu este alcatuita din nici un utilizator, dar mai multi utilizatori se pot alatura echipei, folosind codul secret al acesteia, pentru a concura din partea ei. Este important de notat ca se pot trimite incercari pentru o problema doar de pe conturile proprii, nu de pe echipa in sine. In cazul in care utilizatorii parasesc echipa, punctele generate de acestia se vor pastra.

Fiecare echipa poate alege sa reprezinte o tara, de obicei daca toti membrii sai sunt din aceeasi tara, sau sa nu, astfel fiind o echipa internationala.

Orice tara poate fi reprezentata de mai multe echipe, dar pot exista si tari care inca nu sunt reprezentate de nici o echipa.

La crearea unui concurs acesta nu va avea implicit probleme asociate lui, dar ii se pot adauga multiple probleme.

Persoanele care creeaza probleme sunt numite autori, cand acestia sunt autori noi ei nu au inca nici o problema asociata lor dar nu exista nici o limita la cate probleme pot fi create de ei.

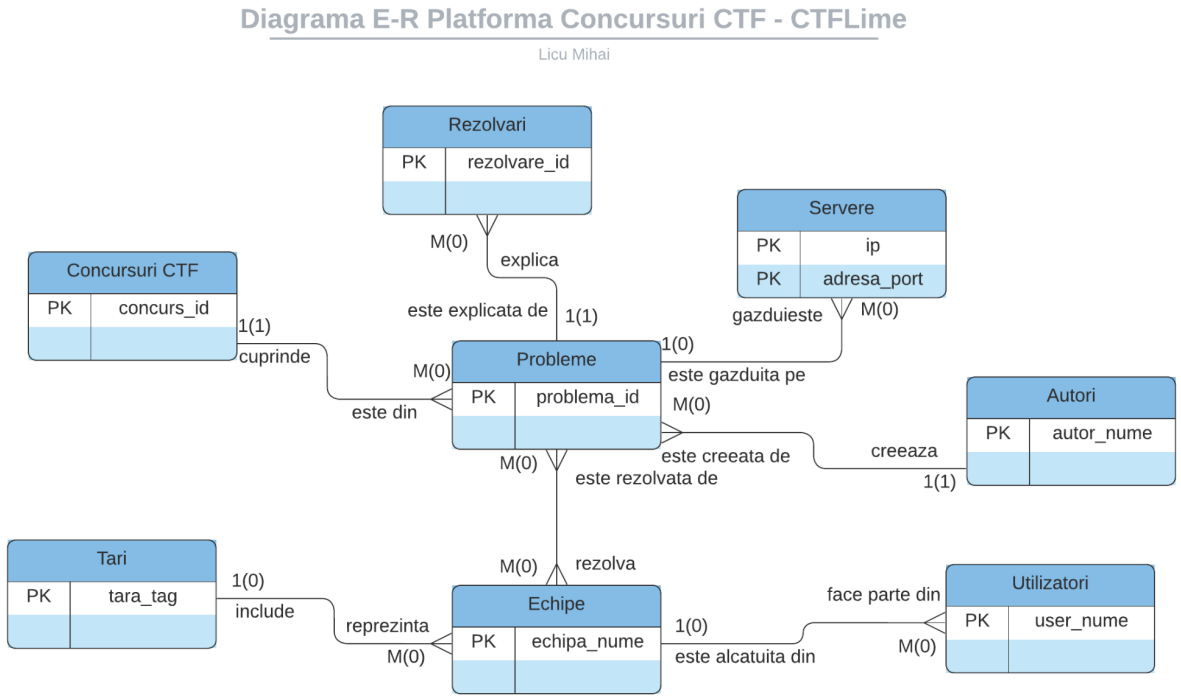
Problemele sunt alcatuite neaparat de exact un singur autor si apartin exact unui singur concurs. De asemenea, unele probleme sunt gazduite pe un server deoarece sunt probleme cu vulnerabilitati ce pot fi exploatate doar remote sau vulnerabilitati web, o astfel de problema poate fi gazduita pe mai multe servere pentru a asigura accesibilitatea in cazul unui influx mare de utilizatori sau al unor probleme tehnice.

Serverele pot fi servere care in momentul acesta gazduiesc o problema sau servere care nu gazduiesc deocamdata nici o problema deci sunt servere "libere".

Rezolvarile sunt documentatii ce explica pas cu pas rezolvarea unei anumite singure probleme. Nu exista rezolvari fara probleme iar unei singure probleme ii pot corespunde mai multe rezolvari.

Orice problema poate fi rezolvata de mai multe echipe, sau posibil in cazul unei probleme foarte grele sa ramana nerezolvata de toata lumea. In timp ce o echipa poate rezolva mai multe probleme sau din pacate nici una.

Diagrama Entitate Relatie



Descrierea entitatilor, atributelor, cheilor, relatiilor si a cardinalitatilor

TARI

Descriere entitate:

Tara reprezinta unitatea geopolitica pe care o echipa o poate reprezenta.

Descriere atribute:

tara_nume - numele reprezentativ si intreg al tarii respective

regiune_tag - regiunea geografica unde se regaseste tara respectiva (am incercat sa urmez standardul goescheme al Natiunilor Unite)

Chei:

tara_tag (PK) - cheie primara pentru TARI, contine cele trei litere conform standardului ISO 3166-1 care reprezinta tara specifica

Relatii si cardinalitati:

Cu tabelul ECHIPE - O tara poate fi reprezentata de mai multe echipe sau nici una. M(0)

ECHIPE

Descriere entitate:

Echipa sub care participa utilizatorii la concursuri.

Descriere atribute:

data_creare - data la care s-a inregistrat aceasta echipa

echipa_email - emailul pe care fi contactact echipa, acest email este public tuturor utilizatorilor de pe platforma dar nu este necesar sa existe

echipa_site - adresa site-ului personal pe care echipa isi publica concursuri, explicatii sau informatii despre ea, public pentru restul utilizatorilor

join_code - un cod alfanumeric de 20 de caractere generat de catre aplicatie la creare unei echipe folosit de utilizatori pentru a intra in acea echipa

academica - determina daca echipa este clasificata ca echipa academica sau nu, pentru unele concursuri doar echipele academice ar fi eligibile pentru premii

Chei:

echipa_nume (PK) - cheie primara pentru ECHIPE, numele unic si nenul care reprezinta echipa

tara_tag (FK) - tara pe care aceasta echipa doreste sa o reprezinte, realizeaza relatia cu tabelul TARI

Relatii si cardinalitati:

Cu tabelul TARI - Prin tara_tag (FK), o echipa poate reprezenta o singura tara sau nici una. 1(0)

Cu tabelul UTILIZATORI - O echipa poate corespunde zero sau mai multor utilizatori. M(0)

Cu tabelul INCERCARI - O echipa poate realiza mai multe incercari pentru mai multe probleme sau zero. M(0)

UTILIZATORI

Descriere entitate:

Reprezinta utilizatorul care va participa la competitii pentru a rezolva probleme.

Descriere atribute:

user_email - email-ul pe care poate fi contactat utilizatorul, folosit si la autentificare, acesta este privat si poate fi vazut doar de catre administratorii site-ului

user_parola - parola secreta a utilizatorului pentru a se autentifica in contul sau

data_creare - data la care a fost creat acest cont de utilizator

Chei:

user_nume (PK) - cheie primara pentru tabelul UTILIZATORI, numele ales de utilizator pentru a il denumi si identifica pe platforma

echipa_nume (FK) - echipa din care utilizatorul face parte, face legatura cu tabelul ECHIPE

Relatii si cardinalitati:

Cu tabelul ECHIPE - Un utilizator poate apartine la un timp unei singure echipe sau a nici unei echipe (cand isi creeaza contul sau cand iese din echipa precedenta). 1(0)

AUTORI

Descriere entitate:

Autorul este persoana care creeaza diversele probleme aflate pe platforma.

Descriere atribute:

autor_email - modul de contact al autor, acesta va fi public pentru utilizatori ca sa poata contacta autorul in privinta cu rezolvarea problemelor sale

companie - in cazul in care un autor contribuie probleme din partea unei companii la care este angajat sau ii este partenera poate specifica acest lucru trecandu-si compania in informatiile sale publice

Chei:

autor_nume (PK) - cheie primara pentru tabelul AUTORI, numele unic al autorului cu care doreste sa fie identificat pe platforma

Relatii si cardinalitati:

Cu tabelul PROBLEME - Nu exista o limita pe cate probleme poate un autor sa contribuie, iar acesta poate sa contribuie si zero probleme in cazul in care este un autor nou fara publicatii. M(0)

CONCURSURI_CTF

Descriere entitate:

Concursul reprezinta competitia alcatuita din probleme la care iau parte utilizatorii, de obicei pentru a castiga premii.

Descriere atribute:

concurs_nume - numele sub care este recunoscut concursul de catre utilizatori

concurs_editie - editia ce reprezinta a cata oara se va tine concursul cu acelasi nume

timp_inceput - timpul cand se va incepe concursul si se vor putea incepe rezolvarile problemelor

timp_terminat - timpul cand se va finaliza concursul si se vor inchide trimiterile incercarilor

Chei:

concurs_id (PK) - cheie primara pentru tabelul CONCURSURI_CTF, identificatorul numeric dat acestui concurs respectiv

Relatii si cardinalitati:

Cu tabelul PROBLEME - Un concurs poate fi nou adaugat si in acest caz el poate sa nu prezinte inca probleme, dar ii se pot fi adaugate un numar nelimitat de probleme. M(0)

PROBLEME

Descriere entitate:

O problema este exercitiul care trebuie rezolvat pentru ca echipa utilizatorului sa primeasca puncte in cadrul concursului.

Descriere atribute:

problema_nume - un nume indicativ pentru a identifica problema

categorie - categoria din care problema face parte, daca o problema este mixta aceasta poate face parte dintr-o singura categorie, cea mai semnificativa pentru rezolvarea sa

flag - o bucata de text (de obicei un sha256 dar poate fi si semnificativa) ascunsa undeva in problema, cu scopul ca utilizatorul sa o gaseasca pentru a dovedi rezolvarea problemei

puncte - numarul de puncte acordate pentru rezolvarea acestei probleme

arhiva - fisierele relevante problemei sub forma unei singure arhive compresate, printre acestea se pot gasi fisiere in care sa existe descrierea problemei(ex: .txt .pdf), fisiere sursa (ex: .c .py), executabile (ex: elf, exe), biblioteci (ex: .dll, .lib, .so) si orice alte fisiere relevante problemei

Chei:

problema_id (PK) - cheie primara pentru tabelul PROBLEME, codul numeric dat acestei probleme prin care este identificata

autor (FK) - autorul care a contribuit aceasta problema, realizeaza legatura cu tabelul AUTORI

concurs_id (FK) - concursul din care aceasta problema face parte, realizeaza legatura cu tabelul CONCURSURI_CTF

Relatii si cardinalitati:

Cu tabelul AUTORI - O problema trebuie sa aiba un autor si poate avea maxim un singur autor. 1(1)

Cu tabelul CONCURSURI_CTF - O problema trebuie neaparat sa apartina unui concurs, iar problemele nu pot fi reutilizate la alte concursuri (rezolvarile lor ar fi deja cunoscute si nu ar avea rost). 1(1)

Cu tabelul SERVERE - Unele, dar nu toate, problemele pot fi accesate remote pe unul sau mai multe servere. M(0)

Cu tabelul REZOLVARI - Problemele pot avea rezolvari care descriu pas cu pas procesul de a le exploata vulnerabilitatile si a ajunge la flag, dar acest lucru nu este necesar, si pot exista mai multe rezolvari diferite pentru fiecare problema. M(0)

Cu tabelul INCERCARI - Fiecare problema poate avea mai multe incercari de la mai multe echipe asupra ei sau nici una. M(0)

REZOLVARI

Descriere entitate:

O rezolvare este descrierea pas cu pas a solutiei unei probleme, avand un scop didactic cu intentia de a prezenta vulnerabilitatile acesteia si cum se pot exploata.

Descriere atribute:

rezolvare_descriere - numele fisierului in care se gaseste rezolvarea propusa problemei respective

oficiala - daca rezolvarea este considerata oficiala sau nu

nota - o nota data rezolvarii care determina calitatea acesteia (0 fiind o calitate joasa, 10 o calitate ridicata)

Chei:

rezolvare_id (PK) - cheie primara pentru tabelul REZOLVARI, cod numeric unic pentru a identifica rezolvarea

problema_id (FK) - problema pe care aceasta rezolvare o descrie, face legatura cu tabelul PROBLEME

Relatii si cardinalitati:

Cu tabelul PROBLEME - O rezolvare trebuie neaparat sa apartina exact unei singure probleme. 1(1)

SERVERE

Descriere entitate:

Un server este masina fizica ce este accesibila la adresa ip descrisa si serveste un serviciu pe un port particular, acest serviciu poate fi o instanta remote a unei probleme, sau un website in cazul unei probleme cu elemente din categoria "web".

Chei:

ip (PK) - in combinatie cu adresa_port defineste cheia primara pentru tabelul SERVERE, reprezinta adresa IPv4 la care se poate gasi o anumita problema

adresa_port (PK) - in combinatie cu ip defineste cheia primara pentru tabelul SERVERE, reprezinta portul pe care aplicatia respectiva asculta pentru trafic si la care se poate realiza conexiunea respectiva

problema_id (FK) - problema care se regaseste gazduita pe acest server, legatura cu tabelul PROBLEME

Relatii si cardinalitati:

Cu tabelul PROBLEME - Un server poate corespunde maxim unei singure probleme in aceelasi timp dar pot exista si servere care nu corespund nici unei probleme, astfel find servere "libere". 1(0)

INCERCARI (TABEL ASOCIATIV)

Descriere entitate:

Aceasta entitate reprezinta tabelul asociativ dintre tabelele PROBLEME si ECHIPE. O incercare este tentativa unei echipe sa rezolve o anumita problema dintr-un concurs.

Descriere atribute:

incercare_flag - bucata de text pe care utilizatorul o trimite pentru aceasta incercare, ca o incercare sa fie corecta acest flag trebuie sa fie egal cu flagul problemei respective incercarii

incercare_timp - momentul la care a fost efectuata incercarea respectiva, ca o incercare sa fie valida timpul acesteia trebuie sa se afle in intervalul timp_inceput si timp_terminat al concursului din care face parte problema

Chei:

incercare_id (PK) - cheia primara pentru tabelul INCERCARI, codul numeric unic asignat unei incercari

echipa_nume (FK) - numele echipei care a efectuat aceasta incercare, realizeaza legatura cu tabelul ECHIPE

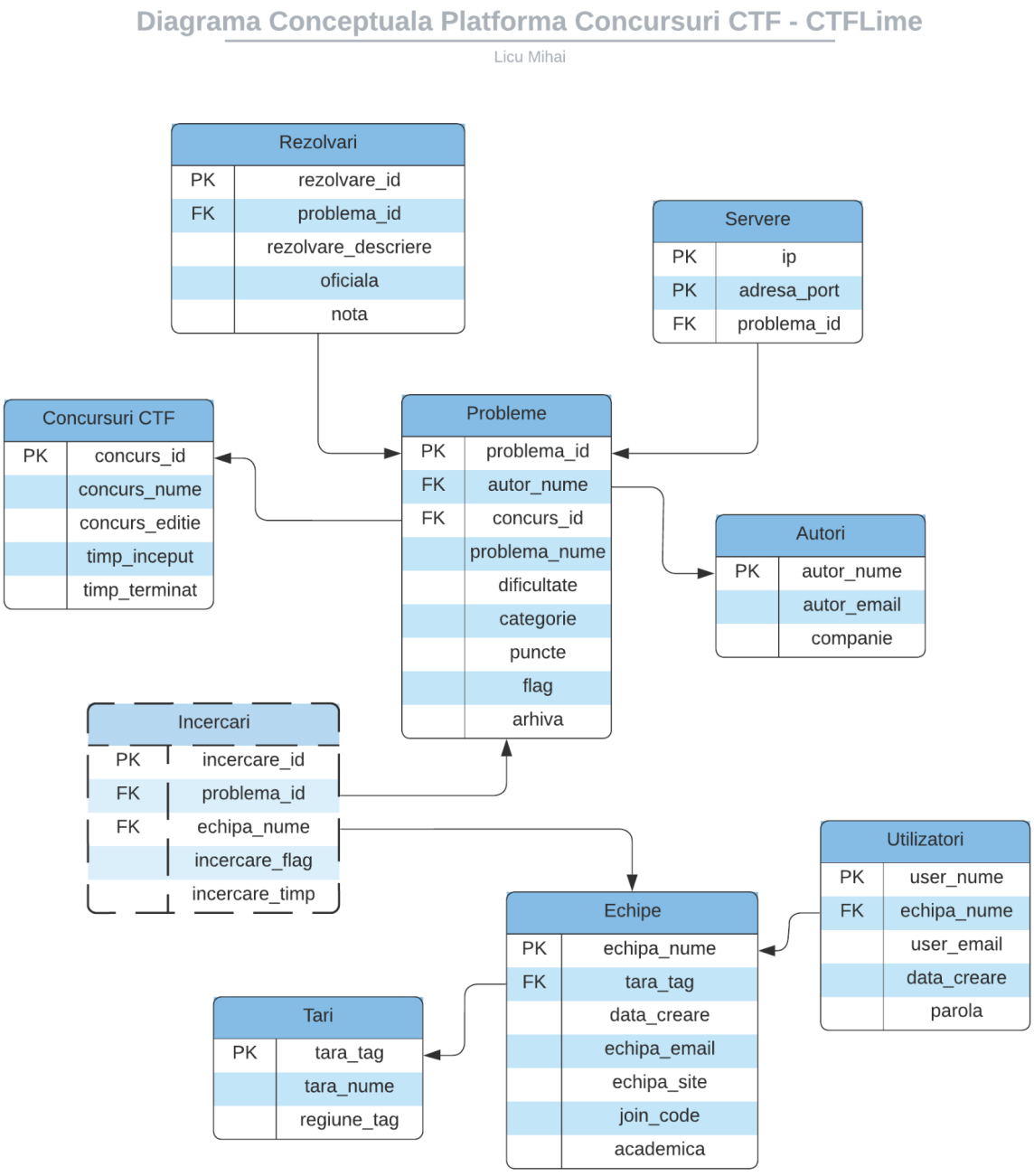
problema_id (FK) - id-ul numeric al problemei careia ii este asociata incercarea, realizeaza legatura cu tabeblul PROBLEME

Relatii si cardinalitati:

Cu tabeul PROBLEME - O incercare corespunde neaparat exact unei probleme. 1(1)
O problema poate avea mai multe incercari sau zero. M(0)

Cu tabelul ECHIPE - O incercare este obligatoriu efectuata de exact o echipa. 1(1)
O echipa poate efectua mai multe incercari sau nici una. M(0)

Diagrama Conceptuala



Descrierea constrangerilor de integritate

TARI

tara_tag VARCHAR(3) - PRIMARY KEY, deoarece urmaresc standardul ISO 3166-1 am implementat un constraint ca acest atribut sa necesite exact 3 caractere CHECK (LENGTH(tara_tag)=3)

tara_nume VARCHAR(25) - UNIQUE deoarece nu pot exista mai multe tari cu aceelasi nume si NOT NULL deoarece nu am putea avea un tag catre o tara inexistenta

regiune_tag VARCHAR(4) - NOT NULL deoarece fiecare tara este inclusa intr-o regiune geografica

ECHIPE

echipa_nume VARCHAR(30) - PRIMARY KEY, un nume dat echipei din maxim 30 caractere

data_creare DATETIME - NOT NULL deoarece doresc a tine cont de timpul de creare al tuturor echipelor pentru statistici viitoare, DEFAULT CURRENT_TIMESTAMP pentru a asigna o valoare default egala cu timpul curent cand s-a creat echipa

echipa_email VARCHAR(40) - UNIQUE deoarece daca ar exista doua echipe contactabile pe aceelasi email s-ar putea transmite informatii private neintentonat,CHECK (LENGTH(user_email)>=6) pentru ca cel mai mai mic email posibil in existenta are 6 caractere (x@x.me), CHECK(echipa_email REGEXP '^[[:alnum:]._%-\+]+@[[:alnum:].-]+[.][[:alnum:]]{2,4}$') acest check asigura ca emailul introdus respecta formatul unui email

echipa_site VARCHAR(40) - UNIQUE ca o echipa sa nu poata utiliza site-ul alteia din motive neprielnice

join_code VARCHAR(20) - UNIQUE pentru ca cu acest cod secret un utilizator se poate alatura unei echipe si dorim ca un cod sa corespunda unei singure echipe, NOT NULL deoarece fiecare echipa are nevoie de un astfel de cod pentru a lasa utilizatori sa intre in ea

academica BOOL - NOT NULL ca sa fie cunoscut statusul fiecarei echipe, DEFAULT FALSE deoarece majoritatea echipelor nu sunt academice si ar usura introducerea datelor

tara_tag VARCHAR(3) - FOREIGN KEY care face legatura cu tabelul TARI

UTILIZATORI

user_nume VARCHAR(25) - PRIMARY KEY, un nume ales de utilizator din maxim 25 caractere si minim 3 caractere CHECK (LENGTH(user_nume)>=3)

user_email VARCHAR(40) - UNIQUE deoarece dorim ca un utilizator sa poata sa isi faca un singur cont per email, NOT NULL deoarece este nevoie de acesta pentru crearea

contului, CHECK (LENGTH(user_email)>=6) pentru ca cel mai mai mic email posibil in existenta are 6 caractere (x@x.me), CHECK(user_email REGEXP '^[[:alnum:]._%-\+]+@[[:alnum:].-]+[.][[:alnum:]]{2,4}$') acest check asigura ca emailul introdus respecta formatul unui email

user_parola VARCHAR(40) - NOT NULL fiecare utilizator are nevoie de o parola pentru a se autentifica in cont, CHECK(LENGTH(user_parola)>=8) din motive de securitate aceasta parola trebuie sa aiba minimul de 8 caractere lungime

data_creare DATETIME - NOT NULL a tine cont de timpul de creare al tuturor utilizatorilor, DEFAULT CURRENT_TIMESTAMP pentru a asigna o valoare default egala cu timpul curent cand s-a creat contul utilizatorului

echipa_nume VARCHAR(30) - FOREIGN KEY care face legatura cu tabelul ECHIPE

AUTORI

autor_nume VARCHAR(25) - PRIMARY KEY, numele ales trebuie sa aiba intre 25 si 3 caractere CHECK (LENGTH(autor_nume)>=3)

autor_email VARCHAR(40) - UNIQUE deoarece dorim ca un autor sa poata fi contactat pe un singur email, NOT NULL pentru ca fiecare autor sa poata fi contactabil, CHECK (LENGTH(autor_email)>=6) pentru ca cel mai mai mic email posibil in existenta are 6 caractere (x@x.me), CHECK(autor_email REGEXP '^[[:alnum:]._%-\+]+@[[:alnum:].-]+[.][[:alnum:]]{2,4}$') acest check asigura ca emailul introdus respecta formatul unui email

companie VARCHAR(20) - compania poate sa aiba maxim 20 de caractere in nume

CONCURSURI_CTF

concurs_id DECIMAL(4) - PRIMARY KEY, CHECK (concurs_id >= 0) ca sa putem stoca 10000 concursuri (id-uri intre 0 si 9999)

concurs_nume VARCHAR(20) - NOT NULL deoarece fiecare concurs are nevoie de un nume care sa il descrie

concurs_editie DECIMAL(3) - NOT NULL deoarece fiecare concurs are o editie asociata acestuia, CHECK (concurs_editie > 0) pentru ca are sens sa avem doar editii strict pozitive, putem avea 999 editii pentru fiecare concurs

timp_inceput DATETIME - NOT NULL deoarece fiecare concurs are nevoie de un punct clar de inceput

timp_terminat DATETIME - NOT NULL deoarece fiecare concurs are nevoie de un punct clar cand se va termina, CHECK (timp_terminat > timp_inceput) ca sa ne asiguram ca concursul se termina dupa ce a inceput ci nu invers

PROBLEME

problema_id DECIMAL(4) - PRIMARY KEY, CHECK (problema_id >= 0) putem avea probleme de la id = 0 pana la id = 9999

problema_nume VARCHAR(20) - NOT NULL fiecare problema are nevoie de un nume pentru a o descrie

categorie VARCHAR(10) - NOT NULL fiecare problema apartine unei categorii

flag VARCHAR(280) - NOT NULL fiecare nevoie are nevoie de un flag altfel nu poate fi rezolvata

puncte DECIMAL(4) - NOT NULL o problema trebuie sa aiba puncte asociate rezolvarii sale ca utilizatorii sa castige puncte in concursul respectiv, CHECK(puncte >= 0) punctele trebuie sa fie strict pozitive (0-9999)

arhiva VARCHAR(20) - NOT NULL in arhiva se afla problema in sine asa ca este nevoie de aceasta altfel nu exista o problema de rezolvat

autor VARCHAR(25) - FOREIGN KEY realizeaza legatura cu tabelul AUTORI, NOT NULL fiecare problema este creata de un autor, nu exista probleme fara autor

concurs_id DECIMAL(4) - FOREIGN KEY realizeaza legatura cu tabelul CONCURSURI_CTF, NOT NULL fiecare problema apartine unui concurs, nu exista probleme din afara concursurilor

CONSTRAINT uq_concurs_numeprob UNIQUE(concurs_id, problema_nume) - la un concurs dorim sa avem o singura problema cu aceelasi nume pentru claritate

CONSTRAINT uq_concurs_arhiva UNIQUE(concurs_id, arhiva) - la un concurs dorim sa avem o singura arhiva cu un nume respectiv pentru a nu confunda arhivele intre probleme

CONSTRAINT uq_numeprob_autor UNIQUE (problema_nume, autor) - dorim ca autorii sa nu reutilizeze numele problemelor sale pentru claritate (cum un autor literar nu ar avea doua carti cu aceelasi nume)

REZOLVARI

rezolvare_id DECIMAL(5) - PRIMARY KEY, CHECK (rezolvare_id >= 0) avem id-uri asignabile de la 0 la 99999, am decis ca acest numar este corect si ar functiona in cazul in care am avea chiar mai multe rezolvari pentru fiecare problema (4 nu ar fi destul)

rezolvare_descriere VARCHAR(35) - NOT NULL nu exista o descriere in sine daca aceasta nu ar avea nume (un fisier asociat ei), CHECK (LENGTH(rezolvare_descriere)>=6) pentru a stabili un nivel de claritate am decis ca o rezolvare sa aiba un numar minim de 6 caractere (4 ar fi ocupate de extensia fisierului)

oficiala BOOL - NOT NULL pentru a face clar pentru fiecare problema daca este oficiala sau nu, DEFAULT FALSE deoarece majoritatea nu vor fi oficiale

nota DECIMAL(4,2) - CHECK(nota BETWEEN 0 and 10) nota acordata trebuie sa fie in intervalul [0.00,10.00]

problema_id DECIMAL(4)- FOREIGN KEY care face legatura cu tabelul PROBLEME, NOT NULL fiecare rezolvare trebuie sa rezolve o problema

CONSTRAINT uq_nume_perproblema UNIQUE(rezolvare_descriere, problema_id) - pentru a putea sa ne referim mai usor la ele, si pentru a le putea stoca eficient (toate rezolvarile pentru aceeasi probleme sa se afle in aceelasi director), dorim sa nu existe nume identice de rezolvare pentru aceeasi problema

SERVERE

ip VARCHAR(20) - CHECK (LENGTH(ip)>=7), cea mai mica adresa IPv4 are 7 caractere lungime (1.1.1.1)

adresa_port DECIMAL(5) - CHECK(adresa_port BETWEEN 1 AND 65535) range-ul posibil pentru porturi (2^{16} optiuni)

CONSTRAINT pk_servere PRIMARY KEY (ip, adresa_port) - PRIMARY KEY-ul acestei entitati este perechea de ip si adresa_port

problema_id DECIMAL(4) - FOREIGN KEY realizeaza relatia cu tabelul PROBLEME

INCERCARI (TABEL ASOCIATIV)

incercare_id DECIMAL(6) - PRIMARY KEY, CHECK (incercare_id >= 0) putem retine 10^6 incercari, ceea ce inseamna ca am putea avea si 100 de incercari per problema

problema_id DECIMAl(4) - FOREIGN KEY realizeaza relatia cu tabelul PROBLEME, NOT NULL fiecare incercare este obligatoriu pentru o anumita problema

echipa_nume VARCHAR(30) - FOREIGN KEY realizaza relatia cu tabelul ECHIPE, NOT NULL fiecare incercare trebuie sa fie facuta de o echipa

incercare_flag VARCHAR(280) - NOT NULL ca o incercare sa poata fi facuta trebuie sa existe un flag asociat acesteia

incercare_timp DATETIME - NOT NULL avem nevoie sa determinam cand a fost facuta incercarea pentru a acorda puncte, DEFAULT CURRENT_TIMESTAMP pentru a asigna valoarea default cand a fost facuta aceasta incercare

CONSTRAINT uq_prob_echipa_timp UNIQUE(problema_id, echipa_nume, incercare_timp) - in cazul in care mai multi utilizatori din aceeasi echipa incearca sa faca incercari concomitent se va accepta cate o singura incercare pentru aceeasi probleme de catre aceeasi echipa la un anumit moment in timp

Descrierea constrangerilor ON DELETE (si ON UPDATE)

Am implementat constrangeri ON DELETE si ON UPDATE pentru fiecare foreign key pentru a definii comportamentul in situatiile respective.

ECHIPE

CONSTRAINT fk_echipe_tara FOREIGN KEY (tara_tag) REFERENCES tari(tara_tag) ON UPDATE CASCADE ON DELETE SET NULL - cand o tara este stearsa din baza de date atunci toate echipele care aveau tag-ul tarii respective vor avea tagul NULL, utilizatorii echipei avand optiunea sa reprezinte alta tara sau sa ramana NULL, cand tag-ul tarii va fi updatat atunci toate echipele care aveau tag-ul tarii respective vor avea noul tag

UTILIZATORI

CONSTRAINT fk_utilizatori_echipa FOREIGN KEY (echipa_nume) REFERENCES echipe(echipa_nume) ON UPDATE CASCADE ON DELETE SET NULL - cand o echipa este stearsa din tabelul ECHIPE toti utilizatorii care apartineau acestei echipe nu vor mai apartine nici unei echipe, dar vor avea optiunea sa intre in alta echipa, cand numele unei echipe este updatat se va updata si pentru fiecare utilizator al acelei echipe

PROBLEME

CONSTRAINT fk_probleme_autor FOREIGN KEY (autor) REFERENCES autori(autor_nume) ON UPDATE CASCADE ON DELETE CASCADE - daca un autor este sters din tabelul AUTORI atunci si toate problemele sale sunt sterse, ON DELETE SET NULL nu este o optiune buna deoarece nu exista probleme fara autori, daca numele sau este schimbat atunci si toate aparitiile sale din probleme vor fi schimbate

CONSTRAINT fk_probleme_concurs FOREIGN KEY (concurs_id) REFERENCES concursuri_ctf(concurs_id) ON UPDATE CASCADE ON DELETE CASCADE - daca un concurs este sters din tabelul CONCURSURI_CTF atunci si toate problemele aferente concursului sunt sterse, ON DELETE SET NULL nu este o optiune buna deoarece nu exista probleme care nu apartin unui concurs, daca concursului ii se schimba id-ul atunci si toate referintele catre acest id se vor schimba

REZOLVARI

CONSTRAINT fk_rezolvari_problema_id FOREIGN KEY (problema_id) REFERENCES probleme(problema_id) ON UPDATE CASCADE ON DELETE CASCADE - daca o problema este stearsa din tabelul PROBLEME atunci se vor sterge si toate rezolvarile acesteia, ON DELETE SET NULL nu este o alegere buna deoarece nu exista rezolvari fara o problema, daca id-ul unei probleme va fi schimbat atunci se va schimba si in toate rezolvarile care refera acea problema

SERVERE

CONSTRAINT fk_servere_problema FOREIGN KEY (problema_id) REFERENCES probleme(problema_id) ON UPDATE CASCADE ON DELETE SET NULL - daca se va sterge o problema din tabelul PROBLEME atunci toate serverele care hostau acea problema vor avea

problema_id NULL si vor deveni servere "libere", daca id-ul unei probleme se va updata atunci se va updata si id-ul problemei in fiecare server unde aceasta era hostata

INCERCARI (TABEL ASOCIATIV)

CONSTRAINT fk_incercari_problema_id FOREIGN KEY (problema_id) REFERENCES probleme(problema_id) ON UPDATE CASCADE ON DELETE CASCADE - daca o problema se va sterge atunci se vor sterge si toate incerarile asociate acesteia, daca id-ul ei se va schimba se va schimba si in toate incercarile ei

CONSTRAINT fk_incercari_echipa_nume FOREIGN KEY (echipa_nume) REFERENCES echipe(echipa_nume) ON UPDATE CASCADE ON DELETE CASCADE - daca o echipa se va sterge atunci se vor sterge si toate incercarile asociate acesteia, daca numele ei se va schimba atunci se va schimba si in toate incerarile ei

ON DELETE SET NULL nu sunt optiuni bune deoarece o incercare trebuie neaparat sa fie al unei echipe pentru o problema

Scriptul SQL

Introducere

Scriptul SQL a fost scris in MySQL Workbench 8.0.27 si a fost salvat sub numele **scriptSQL_LICU_MIHAI_GEORGE_gr262.sql**

Scriptul ruleaza fara probleme inclusiv la prima rulare doarece prezinta check-uri care verifica daca baza de date a fost inca creata sau nu. La rurali successive acesta executa comanda DROP DATABASE si o creeaza din nou.

```
DROP DATABASE IF EXISTS ctflime;
CREATE DATABASE IF NOT EXISTS ctflime;
use ctflime;
```

Tabelul tari

Crearea tabelului tari

```
CREATE TABLE utilizatori
    ( user_nume VARCHAR(25)
    , CONSTRAINT pk_utilizatori PRIMARY KEY(user_nume)
    , CONSTRAINT chk_user_nume_lungime CHECK (LENGTH(user_nume)>=3)
    , user_email VARCHAR(40) UNIQUE NOT NULL
    , CONSTRAINT chk_user_email_lungime CHECK (LENGTH(user_email)>=6)
    , CONSTRAINT regex_user_email CHECK(user_email REGEXP
'^[[:alnum:]._%-\+]+@[[:alnum:].-]+[.][[:alnum:]]{2,4}$')
```

```
            , user_parola VARCHAR(40) NOT NULL
            , CONSTRAINT chk_user_parola_lungime  CHECK(LENGTH(user_parola)>=8)
            , data_creare DATETIME DEFAULT CURRENT_TIMESTAMP NOT NULL
            , echipa_nume VARCHAR(30)
            , CONSTRAINT fk_utilizatori_echipa FOREIGN KEY (echipa_nume) REFERENCES echipe(echipa_nume)
ON UPDATE CASCADE ON DELETE SET NULL
            );
```

Inserari in tabelul tari

```
        INSERT INTO tari(tara_tag, tara_nume, regiune_tag) VALUES("GER", "Germany", "EUW");
        INSERT INTO tari(tara_tag, tara_nume, regiune_tag) VALUES("DEN", "Denmark", "EUW");
        INSERT INTO tari(tara_tag, tara_nume, regiune_tag) VALUES("GBR", "United Kingdom", "EUW");
```

Tabelul echipe

Crearea tabelului echipe

```
        CREATE TABLE echipe
            ( echipa_nume VARCHAR(30)
            , CONSTRAINT pk_echipe PRIMARY KEY(echipa_nume)
            , data_creare DATETIME DEFAULT CURRENT_TIMESTAMP NOT NULL
            , echipa_email VARCHAR(40) UNIQUE
            , CONSTRAINT chk_echipa_email_lungime CHECK (LENGTH(echipa_email)>=6)
            , CONSTRAINT regex_echipa_email CHECK(echipa_email REGEXP
'^[[:alnum:]._%-\+]+@[[:alnum:].-]+[.][[:alnum:]]{2,4}$')
            , echipa_site VARCHAR(40) UNIQUE
            , join_code VARCHAR(20) UNIQUE NOT NULL
            , academica BOOL DEFAULT FALSE NOT NULL
            , tara_tag VARCHAR(3)
            , CONSTRAINT fk_echipe_tara FOREIGN KEY (tara_tag) REFERENCES tari(tara_tag) ON UPDATE
CASCADE ON DELETE SET NULL
            );
```

Inserari in tabelul echipe

```
        INSERT INTO echipe(echipa_nume, echipa_email, tara_tag, echipa_site, join_code, data_creare)
VALUES("dont_thread_on_me",  "dont_thread_on_me@gmail.com", "ROU", "https://github.com/dontthreadonme",
"ae3a42fecbd589bfaca9", STR_TO_DATE("2021-06-21 15:02:02", "%Y-%m-%d %H:%i:%s"));
        INSERT INTO echipe(echipa_nume, echipa_email, tara_tag, echipa_site, join_code, data_creare)
VALUES("WreckTheLine",  NULL, "ROU", "https://wrecktheline.com/", "83fb53b35c6fc7f521a0",
STR_TO_DATE("2017-03-01 11:42:02", "%Y-%m-%d %H:%i:%s"));
        INSERT INTO echipe(echipa_nume, echipa_email, tara_tag, join_code, data_creare)
VALUES("tempname",  "tempmail@gmail.com", "ROU", "b1cba2b81a3184f7f19f", STR_TO_DATE("2019-04-19
11:15:47", "%Y-%m-%d %H:%i:%s"));
```

Tabelul utilizatori

Crearea tabelului utilizatori

```
        CREATE TABLE utilizatori
            ( user_nume VARCHAR(25)
            , CONSTRAINT pk_utilizatori PRIMARY KEY(user_nume)
            , CONSTRAINT chk_user_nume_lungime CHECK (LENGTH(user_nume)>=3)
            , user_email VARCHAR(40) UNIQUE NOT NULL
            , CONSTRAINT chk_user_email_lungime CHECK (LENGTH(user_email)>=6)
            , CONSTRAINT regex_user_email CHECK(user_email REGEXP
'^[[:alnum:]._%-\+]+@[[:alnum:].-]+[.][[:alnum:]]{2,4}$')
            , user_parola VARCHAR(40) NOT NULL
            , CONSTRAINT chk_user_parola_lungime  CHECK(LENGTH(user_parola)>=8)
            , data_creare DATETIME DEFAULT CURRENT_TIMESTAMP NOT NULL
            , echipa_nume VARCHAR(30)
            , CONSTRAINT fk_utilizatori_echipa FOREIGN KEY (echipa_nume) REFERENCES echipe(echipa_nume)
ON UPDATE CASCADE ON DELETE SET NULL
            );
```

Inserari in tabelul utilizatori

```
        INSERT INTO utilizatori(user_nume, user_email, echipa_nume, user_parola, data_creare)
VALUES("leaK.u", "licu.mihai21@gmail.com", "dont_thread_on_me", "03BFqrnPXi", STR_TO_DATE("2021-6-01
13:55:02", "%Y-%m-%d %H:%i:%s"));
        INSERT INTO utilizatori(user_nume, user_email, echipa_nume, user_parola, data_creare)
VALUES("binarysheep", "cristiansimache@gmail.com", "dont_thread_on_me", "zRnqjDUS1C",
STR_TO_DATE("2021-6-01 11:23:45", "%Y-%m-%d %H:%i:%s"));
        INSERT INTO utilizatori(user_nume, user_email, echipa_nume, user_parola, data_creare)
VALUES("Volrin", "maresm@gmail.com", "dont_thread_on_me", "iCvj7D1RdN", STR_TO_DATE("2021-6-15
17:53:54", "%Y-%m-%d %H:%i:%s"));
```

Tabelul autori

Crearea tabelului autori

```
        CREATE TABLE autori
            ( autor_nume VARCHAR(25)
            , CONSTRAINT pk_autori PRIMARY KEY(autor_nume)
            , CONSTRAINT chk_autor_nume_lungime CHECK (LENGTH(autor_nume)>=3)
            , autor_email VARCHAR(40) UNIQUE NOT NULL
            , CONSTRAINT chk_autor_email_lungime CHECK (LENGTH(autor_email)>=6)
            , CONSTRAINT regex_autor_email CHECK(autor_email REGEXP
'^[[:alnum:]._%-\+]+@[[:alnum:].-]+[.][[:alnum:]]{2,4}$')
            , companie VARCHAR(20)
            );
```

Inserari in tabelul autori

```
        INSERT INTO autori(autor_nume, autor_email, companie) VALUES("volf", "volf@gmail.com", NULL);
```

```
        INSERT INTO autori(autor_nume, autor_email, companie) VALUES("Ephvuln", "Ephvuln@gmail.com",
NULL);
        INSERT INTO autori(autor_nume, autor_email, companie) VALUES("yakuhito", "yakuh@gmail.com",
NULL);
```

Tabelul concursuri_ctf

Crearea tabelului concursuri_ctf

```
        CREATE TABLE concursuri_ctf
            ( concurs_id DECIMAL(4)
            , CONSTRAINT pk_concursuri PRIMARY KEY(concurs_id)
            , CONSTRAINT chk_id_concurs_pozitiv CHECK (concurs_id >= 0)
            , concurs_nume VARCHAR(20) NOT NULL
            , concurs_editie DECIMAL(3) NOT NULL
            , CONSTRAINT chk_editie_concurs_strictpozitiv CHECK (concurs_editie > 0)
            , CONSTRAINT uq_nume_editie UNIQUE(concurs_nume, concurs_editie)
            , timp_inceput DATETIME NOT NULL
            , timp_terminat DATETIME NOT NULL
            , CONSTRAINT chk_terminat_dupa_inceput CHECK (timp_terminat > timp_inceput)
            );
```

Inserari in tabelul concursuri_ctf

```
        INSERT INTO concursuri_ctf(concurs_id, concurs_nume, concurs_editie, timp_inceput,
timp_terminat) VALUES(0, "Unbreakable", 1, STR_TO_DATE("2019-03-06 12:00:00", "%Y-%m-%d %H:%i:%s"),
STR_TO_DATE("2019-03-09 12:00:00", "%Y-%m-%d %H:%i:%s"));
        INSERT INTO concursuri_ctf(concurs_id, concurs_nume, concurs_editie, timp_inceput,
timp_terminat) VALUES(1, "Google CTF", 1, STR_TO_DATE("2019-06-08 13:00:00", "%Y-%m-%d %H:%i:%s"),
STR_TO_DATE("2019-06-10 13:00:00", "%Y-%m-%d %H:%i:%s"));
        INSERT INTO concursuri_ctf(concurs_id, concurs_nume, concurs_editie, timp_inceput,
timp_terminat) VALUES(2, "Unbreakable", 2, STR_TO_DATE("2019-10-20 12:00:00", "%Y-%m-%d %H:%i:%s"),
STR_TO_DATE("2019-10-23 12:00:00", "%Y-%m-%d %H:%i:%s"));
```

Tabelul probleme

Crearea tabelului probleme

```
        CREATE TABLE probleme
            ( problema_id DECIMAL(4)
            , CONSTRAINT pk_probleme PRIMARY KEY(problema_id)
            , CONSTRAINT chk_id_problema_pozitiv CHECK (problema_id >= 0)
            , problema_nume VARCHAR(20) NOT NULL
            , categorie VARCHAR(10) NOT NULL
            , flag VARCHAR(280) NOT NULL
            , puncte DECIMAL(4) NOT NULL
            , CONSTRAINT chk_pct_strictpozitive CHECK(puncte >= 0)
            , arhiva VARCHAR(20) NOT NULL
            , autor VARCHAR(25) NOT NULL
```

```
            , CONSTRAINT fk_probleme_autor FOREIGN KEY (autor) REFERENCES autori(autor_nume) ON UPDATE
CASCADE ON DELETE CASCADE
            , concurs_id DECIMAL(4) NOT NULL
            , CONSTRAINT fk_probleme_concurs FOREIGN KEY (concurs_id) REFERENCES
concursuri_ctf(concurs_id) ON UPDATE CASCADE ON DELETE CASCADE
            , CONSTRAINT uq_concurs_numeprob UNIQUE(concurs_id, problema_nume)
            , CONSTRAINT uq_concurs_arhiva UNIQUE(concurs_id, arhiva)
            , CONSTRAINT uq_numeprob_autor UNIQUE (problema_nume, autor)
            );
```

Inserari in tabelul probleme

```
        INSERT INTO probleme(problema_id, concurs_id, autor, categorie, problema_nume, puncte, flag,
arhiva) VALUES(0, 0, "volf", "crypto", "AESbreak", 500,
"CTF{F393AAB2A1C956283A3491EAE53C2875B44B60149A25CD37271102405DB80B00}", "chall.7z");
        INSERT INTO probleme(problema_id, concurs_id, autor, categorie, problema_nume, puncte, flag,
arhiva) VALUES(1, 5, "shad", "crypto", "TWOringMachine", 500,
"CTF{84EBFDFFF45D2DDECB583FFB74E51FDF3F124EC7F0A4D9D7FB2F9EF01B068F31}", "chall.zip");
        INSERT INTO probleme(problema_id, concurs_id, autor, categorie, problema_nume, puncte, flag,
arhiva) VALUES(2, 5, "shad", "web", "xsselsior", 100,
"CTF{DF38371D3332DAFA30B8EEAEEF67F926168FEA403743F0810186E31925A2376A}", "description.rar");
```

Tabelul rezolvari

Crearea tabelului rezolvari

```
        CREATE TABLE rezolvari (
            rezolvare_id DECIMAL(5),
            CONSTRAINT pk_rezolvari PRIMARY KEY (rezolvare_id),
            CONSTRAINT chk_id_rezolvare_pozitiv CHECK (rezolvare_id >= 0),
            rezolvare_descriere VARCHAR(35) NOT NULL,
            CONSTRAINT chk_nume_descriere_lungime CHECK (LENGTH(rezolvare_descriere) >= 6),
            oficiala BOOL NOT NULL DEFAULT FALSE,
            nota DECIMAL(4 , 2 ),
            CONSTRAINT chk_nota_posibila CHECK (nota BETWEEN 0 AND 10),
            problema_id DECIMAL(4) NOT NULL,
            CONSTRAINT fk_rezolvari_problema_id FOREIGN KEY (problema_id)
                REFERENCES probleme (problema_id)
                ON UPDATE CASCADE ON DELETE CASCADE,
            CONSTRAINT uq_nume_perproblema UNIQUE (rezolvare_descriere , problema_id)
        );
```

Inserari in tabelul rezolvari

```
        INSERT INTO rezolvari(rezolvare_id, problema_id, oficiala, nota, rezolvare_descriere) VALUES(0,
1, TRUE, 10, "aesguide.txt");
        INSERT INTO rezolvari(rezolvare_id, problema_id, oficiala, nota, rezolvare_descriere) VALUES(1,
1, FALSE, 7.30, "exploit.txt");
        INSERT INTO rezolvari(rezolvare_id, problema_id, oficiala, nota, rezolvare_descriere) VALUES(2,
4, FALSE, 8.10, "reverseshell.pdf");
```

Tabelul servere

Crearea tabelului servere

```
        CREATE TABLE servere
            ( ip VARCHAR(15)
            , CONSTRAINT chk_ip_lungime CHECK (LENGTH(ip)>=7)
            , adresa_port DECIMAL(5)
            , CONSTRAINT chk_port_range_valid CHECK(adresa_port BETWEEN 1 AND 65535)
            , CONSTRAINT pk_servere PRIMARY KEY (ip, adresa_port)
            , problema_id DECIMAL(4)
            , CONSTRAINT fk_servere_problema FOREIGN KEY (problema_id) REFERENCES probleme(problema_id)
ON UPDATE CASCADE ON DELETE SET NULL
            );
```

Inserari in tabelul servere

```
        INSERT INTO servere(ip, adresa_port, problema_id) VALUES("193.163.64.30", 7659, 1);
        INSERT INTO servere(ip, adresa_port, problema_id) VALUES("1.155.113.185", 3717, 3);
        INSERT INTO servere(ip, adresa_port, problema_id) VALUES("176.178.95.130", 5035, 5);
```

Tabelul incercari (asociativ)

Crearea tabelului incercari

```
        CREATE TABLE incercari
            ( incercare_id DECIMAL(6)
            , CONSTRAINT pk_incercari PRIMARY KEY(incercare_id)
            , CONSTRAINT chk_id_incercare_pozitiv CHECK (incercare_id >= 0)
            , problema_id DECIMAL(4) NOT NULL
            , CONSTRAINT fk_incercari_problema_id FOREIGN KEY (problema_id) REFERENCES
probleme(problema_id) ON UPDATE CASCADE ON DELETE CASCADE
            , echipa_nume VARCHAR(30) NOT NULL
            , CONSTRAINT fk_incercari_echipa_nume FOREIGN KEY (echipa_nume) REFERENCES
echipe(echipa_nume) ON UPDATE CASCADE ON DELETE CASCADE
            , incercare_flag VARCHAR(280) NOT NULL
            , incercare_timp DATETIME DEFAULT CURRENT_TIMESTAMP NOT NULL
            , CONSTRAINT uq_prob_echipa_timp UNIQUE(problema_id, echipa_nume, incercare_timp)
            );
```

Inserari in tabelul incercari

```
        INSERT INTO incercari(incercare_id, problema_id, echipa_nume, incercare_flag, incercare_timp)
VALUES(0, 11, "perfectblue", "CTF{C919C51BDB9494362840C9EF8102C23B22993C908EF7C8A6CD451317E052FEA6}",
STR_TO_DATE("2020-03-09 05:29:37", "%Y-%m-%d %H:%i:%s"));
        INSERT INTO incercari(incercare_id, problema_id, echipa_nume, incercare_flag, incercare_timp)
VALUES(1, 15, "ALLES", "CTF{84FE9B3DB7649546AFAE7B0EA3FF4430104A1184E8BACFD3936300A9961BA9E7}",
STR_TO_DATE("2020-10-27 07:23:57", "%Y-%m-%d %H:%i:%s"));
```

```
        INSERT INTO incercari(incercare_id, problema_id, echipa_nume, incercare_flag, incercare_timp)
VALUES(2, 18, "Katzebin", "CTF{C7055C63375FE984DABB5CB628EE9CDED185C98D37A29019E30247F65117F71E}",
STR_TO_DATE("2019-07-10 15:29:37", "%Y-%m-%d %H:%i:%s"));
        INSERT INTO incercari(incercare_id, problema_id, echipa_nume, incercare_flag, incercare_timp)
VALUES(3, 1, "Katzebin", "CTF{84D4F81C33FD56EA63815C31F7DFBC28C8566757DF9792D6D2E3BF2FD8D5D0C5}",
STR_TO_DATE("2020-07-11 08:30:27", "%Y-%m-%d %H:%i:%s"));
        INSERT INTO incercari(incercare_id, problema_id, echipa_nume, incercare_flag, incercare_timp)
VALUES(4, 8, "tempname", "testflag", STR_TO_DATE("2013-03-06 15:29:37", "%Y-%m-%d %H:%i:%s"));
```