PROIECT BAZE DE DATE

1. Descrierea modelului real, a utilității acestuia și a regulilor de funcționare.

Studenții unei facultăți participă la stagii oferite de către companii. Stagiile pot fi de tip *internship* sau *training*. Stagiile de practică de tip *internship* au fiecare, o dată de început și o dată de final. Stagiile de practică de tip training au câte o dată de început și o dată de final pentru fiecare student admis. Opțional, participarea la unele de stagii de practică poate fi plătită de către companii.

Participarea la stagii de practică în anii terminali este supervizată de un cadru didactic și se finalizează cu acordarea unui singur calificativ (admis/respins), în funcție de adeverința completată de către companie.

Pentru a fi admis la un stagiu de practica de tip training este necesar ca un student să cunoască unele tehnologii, la un anumit nivel cerut de companie.

Pe perioada efectuării unui stagiu studenții pot fi coordonați de specialiști din compania respectivă.

În timpul unui stagiu de tip training compania desfășoară proiecte. Un student poate fi încadrat pe mai multe proiecte din cadrul trainingului, iar în fiecare dintre acestea poate lucra cu mai multe tehnologii (limbaje, sisteme). La finalul stagiului, pentru fiecare student, coordonatorii evaluează nivelul de cunoaștere a tehnologiilor cu care a lucrat studentul, acordând un calificativ pentru fiecare dintre acestea.

2. Prezentarea constrângerilor (restricții, reguli) impuse asupra modelului.

Un cadru didactic poate superviza zero sau mai mulți studenți.
Fiecare student primește un calificativ admis sau respins.
O companie poate organiza unul sau mai multe stagii de practică.
Un student poate avea în cadrul stagiului, mai mulți coordonatori.
Un coordonator poate coordona în cadrul unui stagiu, mai mulți studenți.
etc.

3. Descrierea entităților, incluzând precizarea cheii primare.

ENTITATE	**CHEIE PRIMARA**	**OBSERVATII**
stagiu	stagiu_id	Informații generale despre stagii propuse de companii
internship	stagiu_id	Informații specifice despre stagii de tip internship
tehnologie	tehnologie_id	Tehnologie se refera la limbaje de programare, cunostente de arhitectura sistemelor, framework-uri, baze de date etc.
specialist	specialist_id	Specialiștii sunt persoane, angajați ai unei companii

4. Descrierea relațiilor, incluzând precizarea cardinalității acestora.

RELATIE	CARDINALITATE	OBSERVATII
coordoneaza	specialist-(sudent, stagiu) one-to-many student-(specialist,stagiu) one-to-many Stagiu-(speciaist, student) one-to-many	Un student poate avea mai mulți coordonatori pe parcursul unui stagiu
participa	Student-stagiu many-to-many	Un student poate participa la mai multe stagii. La un stagiu pot participa mai mulți studenți.

etc.

5. Descrierea atributelor, incluzând tipul de date și eventualele constrângeri, valori implicite, valori posibile ale atributelor.

ENTITATE: SUPERVIZEAZA

Atribut	Tip	Dimensiune/ precizie	Valori posibile si valori default	Observații, obligatoriu/opțional
calificativ	string	6	admis, respins	

ENTITATE: STAGIU

Atribut	Tip	Dimensiune/ precizie	Valori posibile si valori default	Observații, obligatoriu/opțional
data_inceput	date			
data_final	date			
Sesiune_online_ prezentare	string			Link-uri către întâlniri online de prezentare a programului de stagiu. Atribut multiplu

RELATIE: PARTICIPA

Atribut	Tip	Dimensiune/ precizie	Valori posibile si valori default	Observații, obligatoriu/opțional
data_inceput	date			Data la care un student începe stagiul, NOT NULL
data_final	date			Data la care un student finalizează stagiul,
salariu	float	6	Valoare default 0	Suma plătită de companie pentru participare la stagiu, NULL

RELATIE: CUNOSTINTE_NECESARE

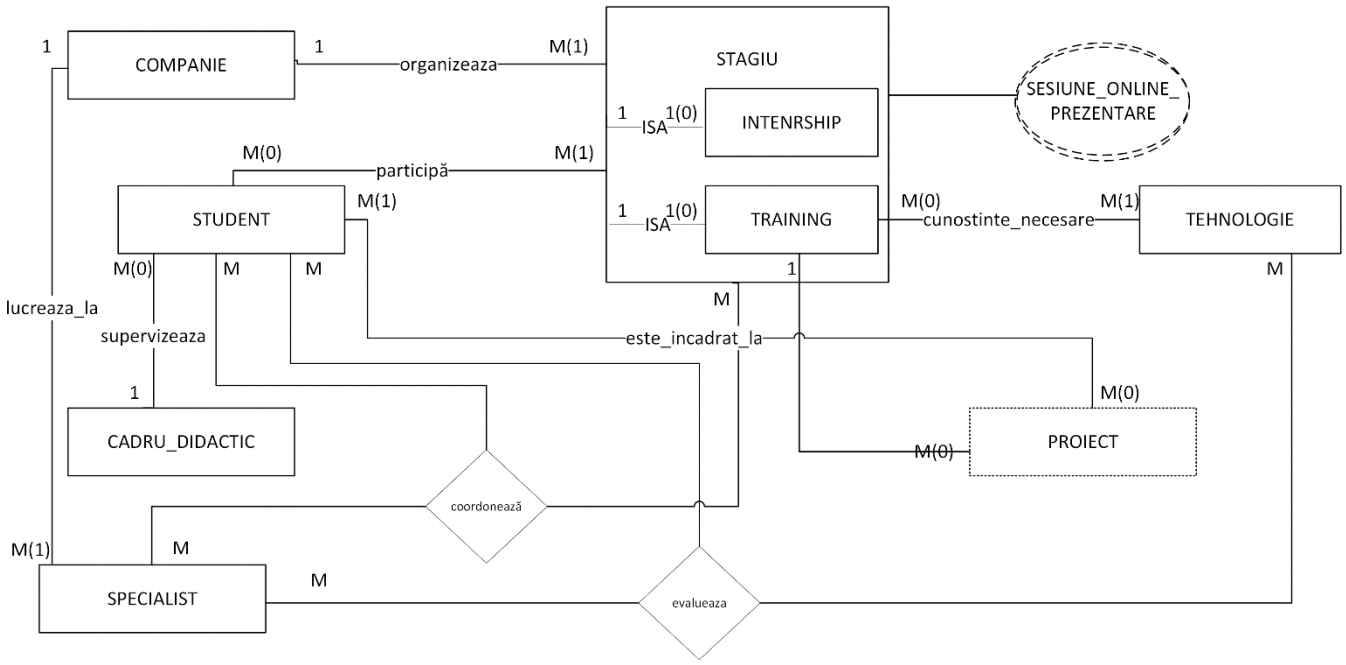
Atribut	Tip	Dimensiune/ precizie	Valori posibile si valori default	Observații, obligatoriu/opțional
nivel	integer	1	[1,5]	Nivelul la care un candidat trebuie să cunoască o tehnologie pentru a fi admis, NOT NULL

NTITATE: SUPERVIZEAZA

Atribut	Tip	Dimensiune/ precizie	Valori posibile si valori default	Observații, obligatoriu/opțional
nivel	integer	1	[1,5]	Nivelul la care un candidat cunoaște o tehnologie, NOT NULL

etc.

6. Realizarea diagramei entitate-relație corespunzătoare descrierii de la punctele 3-5.



7. Realizarea diagramei conceptuale corespunzătoare diagramei entitate-relație proiectate la punctul 6. Diagrama conceptuală obținută trebuie să conțină minimum 6 tabele (fără considerarea subentităților), dintre care cel puțin un tabel asociativ.

8. Enumerarea schemelor relaționale corespunzătoare diagramei conceptuale proiectate la punctul 7.

   COMPANIE(companie_id#, denumire, adresa, site_web)

   STUDENT(student_id#, nume, prenume, nr_matricol, data_nasterii)

   COORDONEAZA(specialist_id#, sudent_id#, stagiu_id#, feedback_specialist)

   EVALUEAZA(specialist_id#, student_id#, tehnologie_id#, nivel, data_evaluare)

   etc.

9. Realizarea normalizării până la forma normală 3 (FN1-FN3).

   Presupunem că tabelul corespunzător participării dezvoltatorilor la proiectele unei companii are schema relațională PARTICIPARE (cod_student, nume_student, cod_proiect, denumire_proiect, cod_sef_proiect, nume_sef_proiect, buget, data_inceput_proiect, data_limita_proiect, cod_task, descriere_task, data_incepere_task, durata_realizare_task).

   a) Determinarea mulțimii dependențelor F funcționale care există între atributele acestei relații.

      (cod_student) → (nume_student)
      (cod_proiect) → (denumire_proiect)
      (cod_proiect) →(cod_sef_proiect)
      (cod_sef_proiect) → (nume_sef_proiect)
      (cod_proiect) → (buget)
      (cod_proiect) → (data_inceput_proiect)
      (cod_proiect) → (data_limita_proiect)
      (cod_task) → (descriere_task)
      (cod_task) → (data_incepere_task)
      (cod_task) → (cod_proiect)
      (cod_student, cod_task) → (cod_proiect, durata_realizare_task)

   b) Aduceți relația la forma normală 3, justificând transformările care au loc la fiecare pas. (prin descompuneri fără pierderi de infamații, casey-delobel)
   c) Normalizați PARTICIPARE în FN3 utilizând algoritmul de sinteză.

      Calculare forma canonica a lui F_c:
      $F = F_c$
      Repetă

      - Union rule: se înlocuiesc $\alpha_1 \to \beta_1$ și $\alpha_1 \to \beta_2$ cu $\alpha_1 \to \beta_1\beta_2$

      - Se elimină atribute din membrul stâng sau din membrul drept al unei dependențe:

        Exemplu F = $\{\alpha_1\alpha_2 \to \beta_2, \alpha_1 \to \beta_1, \beta_1 \to \beta_2\}$ se poate elimina α_2 din $\alpha_1\alpha_2 \to \beta_2$
        F = $\{\alpha_1 \to \beta_2, \alpha_1 \to \beta_1, \beta_1 \to \beta_2\}$

        Exemplu F = $\{\alpha_1\beta_1 \to \alpha_2\beta_2, \alpha_1 \to \alpha_2\}$ se poate elimina α_2 din $\alpha_1\beta_1 \to \alpha_2\beta_2$

F = $\{\alpha_1\beta_1 \rightarrow \beta_2, \alpha_1 \rightarrow \alpha_2\}$

Până când mulțimea F nu se modifică

Algoritmul de sinteză
Repetă

- Calculare F_c
- Pentru fiecare dependență $\alpha \rightarrow \beta$ din F_c se adaugă o relație $R_i = \alpha, \beta$
- Dacă nicio relație R_i nu conține o cheie candidat, se adaugă o relație $R_k = K$, K cheie candidat pentru R
- Dacă există *i, j* astfel încât $R_i \subseteq R_j$, se elimină R_i.

Până când mulțimea relațiilor $\{R_i\}$ nu se modifică

10. SQL