Temă de casă

Proiectarea Bazelor de Date

Nicoi Alexandru
Grupa 353
Facultatea de Matematică-Informatică
Universitatea din București

Cuprins

Exercițiul 1

Cerința a)

Să se dea un exemplu de atribut repetitiv (multivaloare) al unei entități în modelul entitate-legătură.

Un exemplu de atribut multivaloare ar fi următorul: în cazul unei baze de date cu locomotive din cadrul unui depou, considerăm entitatea REVIZII, unde stocăm pentru o locomotivă data reviziei efectuată semestrial. Considerăm atributul "data_revizie" care poate lua 0 (dacă locomotiva este nouă) sau mai multe valori, în funcție de vechimea acesteia.

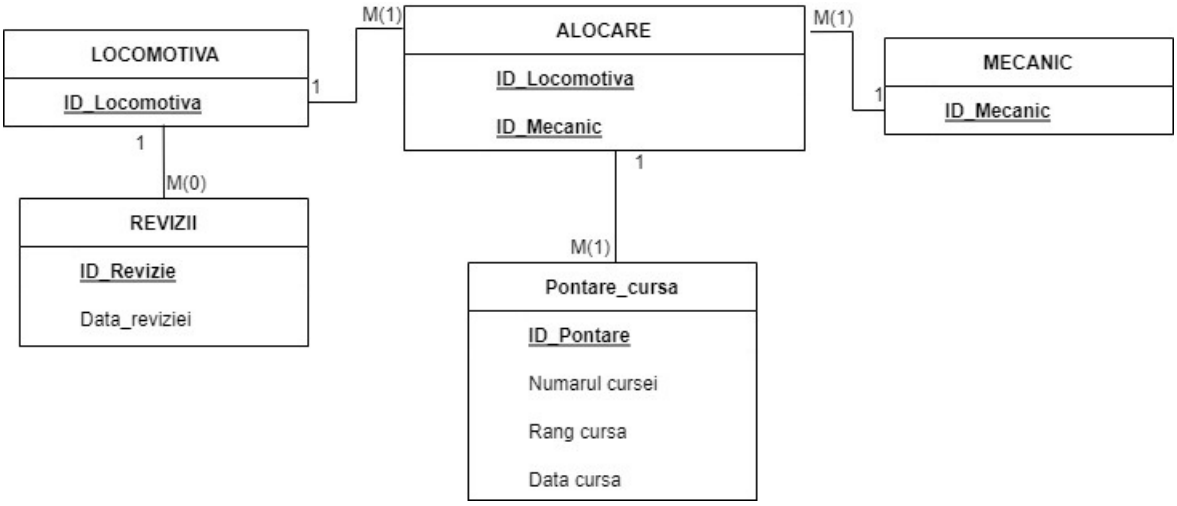
Cerința b)

Să se dea un exemplu de atribut repetitiv (multivaloare) al unei relații mulți-la-mulți în modelul entitate-legătură.

Un exemplu de atribut multivaloare ar fi următorul: în același caz precum cel enunțat mai sus, considerăm entitățile LOCOMOTIVĂ și MECANIC. O locomotivă poate fi condusă de mai mulți mecanici, iar un mecanic poate conduce mai multe locomotive. Atributul multivaloare este reprezentat de numărul cursei pe care o face (spre exemplu - locomotiva cu numărul de parc 692 - condusă de mecanicul Ion Popescu - poate efectua cursa IR 472 în data de 02.01.2022, dar se poate întâmpla ca în data de 04.01.2022 să efectueze cursa IR 361.)

Cerința c)

Să se arate cum se transformă atributele de mai sus la crearea design-ului logic al unei baze de date relaționale.



Pentru a realiza specificația enunțată la a), am pornit cu două entități - LOCOMOTIVA și REVIZII - existența atributului multivaloare fiind confirmată de relația one-to-many, unde pentru o locomotivă pot exista mai multe înregistrări în tabelul REVIZII.

Pentru a realiza specificația enunțată la b), am creat relația many-to-many între LOCOMOTIVA și MECANIC, folosind tabelul asociativ denumit ALOCARE. Folosindu-ne de existența acestui tabel care ne leagă cele două chei din cele două tabele, am creat tabelul Pontare_cursa unde se vor regăsi atributele multivaloare care definesc cursa - numărul, rangul și data. În cazul

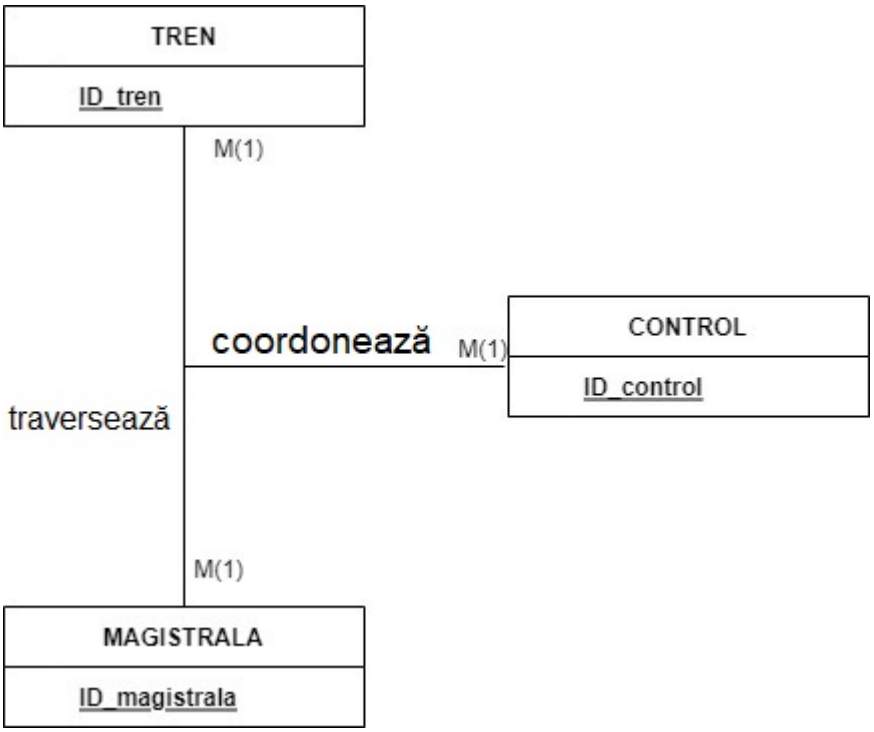
nostru, folosind relația one-to-many - pentru o înregistrare a relației many-to-many, vor exista diverse pontări de curse unde pot exista curse cu alt număr, rang și dată.

Exercițiul 2

Cerința a)

Să se dea un exemplu de relație de tip 3 (între mai mult de două entități) în modelul entitate-legătură.

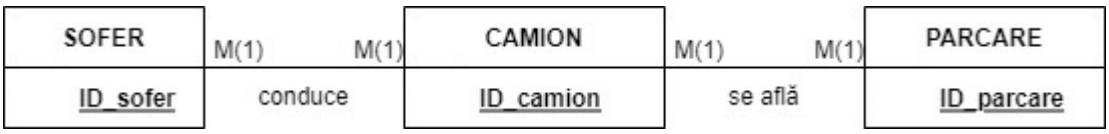
O relație de tip 3 poate fi una reprezentată de organizarea rețelei feroviare. Un tren poate circula pe mai multe magistrale, iar o magistrală poate fi traversată de mai multe trenuri. O magistrală poate fi coordonată/ administrată de mai multe controale (IDM - împiegați de mișcare), iar un control poate coordona mai multe magistrale. De asemenea, și un tren poate fi coordonat de mai multe controale, iar un control poate coordona mai multe trenuri.



Cerința b)

Să se dea un exemplu de trei sau mai multe entități care nu formează o relație de tip 3, ci, relația aparentă de tip 3, "se sparge" de fapt în relații mulți-la-mulți (între câte două entități).

O relație de tip 3 poate fi reprezentată de activitatea unei parcări de camioane (unde șoferii fac pauzele obligatorii). O parcare poate avea mai multe camioane, iar un camion se poate opri la mai multe parcări. De asemenea un camion poate fi condus de mai mulți șoferi, iar un șofer poate conduce mai multe camioane. Deși într-o parcare pot dormi mai mulți șoferi, iar un șofer poate dormi în mai multe parcări, acest lucru reiese din relația sa cu entitatea CAMION deoarece un camion nu poate ajunge în parcare fără șofer (sau chiar să rămână fără șofer). De aceea este o relație aparentă de tip 3, ea fiind de fapt o succesiune de relații many-to-many.



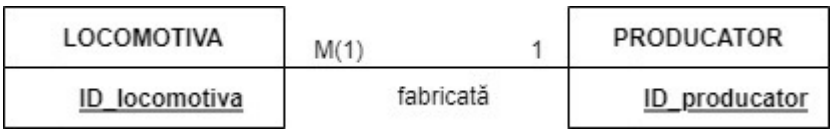
Exercițiul 3

Cerința a)

Să se dea un exemplu de tabel relațional care este în FN1, dar nu în FN2. Să se aducă tabelul în FN2.

Tabelul LOCOMOTIVA prezintă informații despre producător (nume_producator, oras_producator, anul_fondarii), și detalii tehnice (clasa, numar_parc, putere).

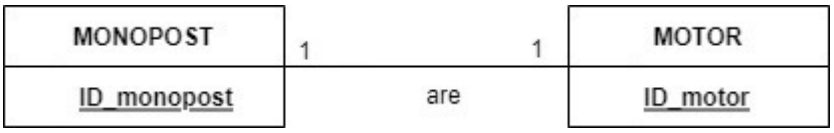
Din construcția de mai sus, se poate observa cheia primară compusă din nume_producator, clasa și numar_parc. Deoarece clasa și numărul de parc trebuie să fie unice, pot exista însa locomotive făcute de același producător, iar câmpurile oras_producator si anul_fondarii depind de producător. Pentru a aduce tabelul în FN2, vom sparge în două tabele sub forma următoare:



Cerința b)

Să se dea un exemplu de tabel relațional care este în FN2, dar nu în FN3. Să se aducă tabelul în FN3.

Tabelul MONOPOST are detalii despre o mașină de Formula 1. Putem avea următoarele atribute - id_monopost, id_echipa, cod_motor, nume_motor. Se poate vedea dependența tranzitivă deoarece numele motorului nu este legat direct de mașină, ci indirect prin codul motorului. Astfel, pentru a aduce tabelul în FN3, vom sparge în două tabele sub forma următoare:



Exercițiul 4

Să se dea un exemplu de tabel relațional în care există o dependență multivaloare (multidependență) între atributele (coloanele) sale, care nu este dependență funcțională.

Considerăm următorul tabel:

PILOT - ID_pilot, ID_masina, ID_costum

Deoarece nu există nicio corelație directă între ID_masina și ID_costum, iar acestea nici nu depind funcțional de ID_pilot, este o dependență multivaloare.

Exercițiul 5

Cerința a)

Să se ilustreze printr-un exemplu structura unui index de tip arbore B* și modul în care o interogare SQL folosește acest index.

Indexul de tip arbore B* se folosește pentru a realiza o căutare secvențială, fiind mai eficientă din punctul de vedere al execuției query-ului pe câmpul unde-l plasăm.

Atunci când facem un query într-un tabel, iar în where avem o coloană pe care am setat indexul, va face prin acel index căutarea, astfel se va crește viteza execuției query-ului.

Considerăm tabelul PRODUCATOR cu prod_id cheie primară, și nume_producator.

```
create index idx_prod_nume on producator(nume_producator);

select prod_id from producator where lower(nume_producator)='softronic
    ';
```

Cerința b)

Să se ilustreze printr-un exemplu structura unui index de tip bitmap și modul în care o interogare SQL folosește acest index.

Spre deosebire de indexul de mai sus, indexul de tip bitmap se folosește dacă coloana pe care se plasează nu are valori foarte variate.

În cazul nostru, considerând tabela LOCOMOTIVA, cu cheia primară id_loco, și coloana PROPULSIE. Un tren poate fi electric, diesel-electric, diesel-hidraulic, abur. Fiindcă avem doar 4 variante, plasând bitmap index pe această coloană, va crește viteza de execuție pentru că va găsi rapid valoarea căutată.

```
create bitmap index idx_loco_propulsie on locomotiva(propulsie);

select id\_loco from locomotiva where lower(propulsie)='diesel';
```

Exercițiul 6

Să se ilustreze printr-un exemplu modul în care o vedere (vizualizare) poate fi folosită pentru a asigura securitatea într-o bază de date.

O vizualizare poate fi folosită pentru a nu permite accesul direct la datele din tabele, creând astfel posibilitatea pentru un user cu privilegii scăzute să primească acces doar pe o bucată din setul de date mare al bazei de date.

Ca exemplu, putem crea un view care permite unui mecanic de locomotivă sa vadă doar locomotivele conduse de el.

```
CREATE OR REPLACE VIEW locomotivele_conduse as
select l.clasa "Clasa", l.numar_parc "Numar de parc"
from locomotiva l join alocare a using(id_locomotiva)
where a.user_mecanic=USER;
```

Exercițiul 7

Să se arate printr-un exemplu în ce condiții este posibil ca două select-uri identice consecutive (fără nici o altă comandă între ele), efectuate în aceeași sesiune de lucru, pe același tabel, pot produce rezultate diferite.

Considerăm tabelul MASINA cu câmpurile - id_masina, marca, model, data_adaugarii.

data_adaugarii va fi un atribut de tip timestamp pentru a memora complet timpul. Deoarece nu putem avea comenzi între select-uri, timpul reprezintă variabilă mobilă, astfel ne vom folosi de SYSTIMESTAMP. Vom cere să afișeze doar record-urile pentru care secundele din diferența între timpul curent și data adăugirii se află între 10 și 20.

```
--Creare tabel--
CREATE TABLE masina(
    id_masina int constraint pk_masina primary key,
    marca varchar(50) not null,
    model varchar(50) not null,
    data_adaugarii timestamp not null,
    constraint uq_marca_model unique (marca,model)
    );

--Inserare tabel--
insert into masina values(1,'Dacia','Logan',to_timestamp('08-01-2022
   15:00:00', 'dd-mm-yyyy hh24:mi:ss'));
insert into masina values(2,'Dacia','Supernova',to_timestamp('
   08-01-2022 15:00:10', 'dd-mm-yyyy hh24:mi:ss'));
insert into masina values(3,'Renault','Megane',to_timestamp('
   08-01-2022 15:00:20', 'dd-mm-yyyy hh24:mi:ss'));
insert into masina values(4,'Renault','Clio',to_timestamp('08-01-2022
   15:00:30', 'dd-mm-yyyy hh24:mi:ss'));
insert into masina values(5,'Citroen','C4',to_timestamp('08-01-2022
   15:00:40', 'dd-mm-yyyy hh24:mi:ss'));

--Query-ul--
select *
from masina
where extract(second from (systimestamp - data_adaugarii)) > 10 and
   extract(second from (systimestamp - data_adaugarii)) < 20;
```

Exercițiul 8

Să se dea un exemplu care să ilustreze interblocarea.

Pentru a ilustra interblocarea, considerăm următorul exemplu:

```
-- Prima tranzactie --
update masina
set model='Logan2'
where id=1;

update circuit
set zile_desfasurare=3
where id=4;

-- A doua tranzactie --
update circuit
set zile_desfasurare=2
where id=3;

update masina
```

```
set model='Nova'
where id=2;
```

În cadrul primei tranzacții, se blochează tabelul masina deoarece se realizează operația de update, analog pentru circuit în cea de-a doua tranzacție. Atunci când încearcă sa treacă mai departe se întamplă fenomenul de interblocare deoarece cele două sunt blocate așteptându-se una pe cealaltă, până când SGBD-ul sesizează această problemă și o remediază.

Exercițiul 9

Să se ilustreze printr-un exemplu utilizarea unui trigger pentru a realiza o constrangere de integritate care nu ar putea fi implementată folosind un constraint din definiția unui tabel.

Putem realiza un trigger prin care să nu permitem schimbarea puterii unui motor cu mai puțin de 30 cai putere.

```
create or replace trigger mytrig
before update of putere on masina
for each row
begin
    if :new.putere <= :old.putere then
     raise_application_error(-20000, 'Valoarea puterii este mai mica sau egala decat cea originala!');
      end if;
      if :new.putere > :old.putere then
         if (:new.putere - :old.putere) < 30 then
             raise_application_error(-20001, 'Valoarea puterii este mai mare dar insuficienta (sub 30 cai putere)');
         else
             dbms_output.put_line('Ai imbunatatit motorul cu ' || :new.putere);
         end if;
    end if;
end;
```

Exercițiul 10

Să se ilustreze printr-un exemplu de program PL/SQL multi-bloc modul de propagare a excepțiilor. Vor fi ilustrate cel putin situațiile în care o excepție este tratată sau nu în blocul curent și în care controlul programului va fi transmis blocului următor din secvență sau blocului exterior.

Considerăm următorul tabel:

ID_MASINA	MARCA	MODEL	DATA_ADAUGARII	PUTERE
1	Dacia	Logan	08-01-2022 15:00:00,000000000	85
2	Dacia	Supernova	08-01-2022 15:00:10,000000000	70
3	Renault	Megane	08-01-2022 15:00:20,000000000	130
4	Renault	Clio	08-01-2022 15:00:30,000000000	90
5	Citroen	C4	08-01-2022 15:00:40,000000000	85

```
begin
    declare
        my_car masina%rowtype;
```

```
4     begin
5         select * into my_car from masina where marca='Dacia';
6         EXCEPTION
7             when TOO_MANY_ROWS then
8                 dbms_output.put_line('Sunt mai multe masini marca
    Dacia -- aici e blocul 1');
9     end;
10
11    declare
12        type masina_record is record(
13            nume_marca masina.marca%type,
14            nume_model masina.model%type);
15        my_car masina_record;
16    begin
17        select marca, model into my_car from masina
18        where putere < 100;
19        EXCEPTION
20            when NO_DATA_FOUND then
21                dbms_output.put_line('Nu sunt masini mai slabe de 100
    cai putere -- aici e blocul 2');
22    end;
23
24    declare
25        type masina_record is record(
26            nume_marca masina.marca%type,
27            nume_model masina.model%type);
28        my_car masina_record;
29    begin
30        select marca, model into my_car from masina
31        where putere < 100;
32        EXCEPTION
33            when TOO_MANY_ROWS then
34                dbms_output.put_line('Nu sunt masini mai slabe de 100
    cai putere -- aici e blocul 3');
35    end;
36
37 EXCEPTION
38    when TOO_MANY_ROWS then
39        dbms_output.put_line('Nu sunt masini mai slabe de 100 cai
    putere -- aici e blocul extern');
40 end;
```

Rezultatul:

```
Sunt mai multe masini marca Dacia -- aici e blocul 1
Nu sunt masini mai slabe de 100 cai putere -- aici e blocul extern


PL/SQL procedure successfully completed.
```

În cadrul primului bloc, dorim să salvăm în variabila my_car recordul cu marca Dacia. În cazul tabelului nostru, avem două recorduri cu marca DACIA, prin urmare va face throw de TOO_MANY_ROWS, eroare pe care o preluăm la linia 7. Pentru că am tratat excepția, se trece la al doilea bloc unde dorim să stocăm mașina care are mai puțin de 100 cai putere. În cazul de față, avem 4 mașini cu această caracteristică si va face din nou throw de TOO_MANY_ROWS, însă nu am tratat eroarea. Chiar dacă în următorul bloc (3) am tratat eroarea, fiind practic

același query și aceeași cerință, nu va intra pe acel bloc, ducându-se în blocul extern definit la linia 1, astfel va ajunge la linia 38 unde va executa instrucțiunea de la linia 39.

Exercițiul 11

Să se ilustreze prin exemple folosirea instrucțiunii RAISE pentru a ridica atât o excepție predefinită cât și o excepție definită de utilizator. În cazul excepțiilor predefinite, să se explice cum anume folosirea instrucțiunii RAISE schimbă funcționalitatea programului (față de cazul când această instrucțiune nu există).

Considerăm următorul tabel:

ID_MASINA	MARCA	MODEL	DATA_ADAUGARII	PUTERE
1	Dacia	Logan	08-01-2022 15:00:00,000000000	85
2	Dacia	Supernova	08-01-2022 15:00:10,000000000	70
3	Renault	Megane	08-01-2022 15:00:20,000000000	130
4	Renault	Clio	08-01-2022 15:00:30,000000000	90
5	Citroen	C4	08-01-2022 15:00:40,000000000	85

```
1  ---------- RAISE EXCEPTIE PREDEFINITA
2  declare
3      type masina_record is record(
4          nume_marca masina.marca%type,
5          nume_model masina.model%type);
6      my_car masina_record;
7  begin
8      select marca, model into my_car from masina
9      where marca='Citroen';
10     raise NO_DATA_FOUND;
11     dbms_output.put_line(my_car.nume_marca || ' ' || my_car.nume_model
       );
12     EXCEPTION
13         when NO_DATA_FOUND then
14             dbms_output.put_line('Nu exista nicio masina marca Citroen
       ');
15         when TOO_MANY_ROWS then
16             dbms_output.put_line('Sunt mai multe masini marca Citroen'
       );
17 end;
18
19 ---------- RAISE EXCEPTIE CUSTOM
20 declare
21     type masina_record is record(
22         nume_marca masina.marca%type,
23         nume_model masina.model%type);
24     type colectie is table of masina_record INDEX BY BINARY_INTEGER;
25     my_colectie colectie;
26     exceptia_mea EXCEPTION;
27 begin
28     select marca, model bulk collect into my_colectie from masina
29     where marca='Dacia';
30     if my_colectie.count < 2 then
31         raise exceptia_mea;
32     else
33         dbms_output.put_line('Ai ' || my_colectie.count || ' masini
       marca Dacia');
34     end if;
35     EXCEPTION
36         when NO_DATA_FOUND then
```

```
37              dbms_output.put_line('Nu exista nicio masina marca Dacia')
    ;
38          when exceptia_mea then
39              dbms_output.put_line('Ai doar o masina Dacia');
40  end;
```

În cazul excepției predefinite, având linia 10 necomentată, în momentul în care va ajunge la ea va face throw-ul acelei erori și ne va duce în zona de EXCEPTION, unde va fi prinsă la linia 13, chiar dacă noi avem mașină marca Citroen. Dacă comentăm linia, ne va afișa mașina memorată în cadrul selectului în variabila my_car. Dacă vom șterge recordul Citroen, chiar dacă linia este comentată, tot va face throw de acea eroare deoarece nu s-au găsit date care să fie transmise în variabila definită. Dacă în schimb avem 2 sau mai multe mașini marca Citroen, indiferent de starea liniei 10 (comentată/necomentată), va face throw de TOO_MANY_ROWS, eroare prinsă la linia 15.

```
Nu exista nicio masina marca Citroen


PL/SQL procedure successfully completed.

Citroen C4


PL/SQL procedure successfully completed.
```

În cazul excepției definită de utilizator, aceasta este declarată la linia 26 numită exceptia_mea, eroare care apare atunci când în tabelul my_colectie avem mai puțin de 2 valori, lucru verificat la linia 30. Dacă se întâmplă vom face throw de excepția custom care va fi preluată la linia 38. Însă dacă nu ne va returna nimic select-ul, va apărea eroarea NO_DATA_FOUND care este și ea prelucrată la linia 36.

```
Ai 2 masini marca Dacia


PL/SQL procedure successfully completed.


1 row deleted.

Ai doar o masina Dacia


PL/SQL procedure successfully completed.
```