Proiect Baze de Date

Stație de colectare și sortare a deșeurilor

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

O importantă problemă comunitară, în ziua de astăzi, o constituie gestionarea deșeurilor. Astfel, a apărut necesitatea existenței stațiilor de colectare și sortare de deșeurilor, în vederea reciclării celor care oferă acest beneficiu. Deșeurile pot fi de mai multe tipuri, fiind selectate pe sortimente, culori şi compoziţie.

Procesul tehnologic constă în sortarea mecanică şi manuală a deşeurilor colectate selectiv. Deşeurile sunt balotate pe categorii și livrate centrelor care le reciclează (reprezentând clienții). Furnizorii pot fi atât persoane fizice, cât și juridice: restaurante, hoteluri, magazine, instituții publice etc., dar și coșurile de gunoi din orașe, dotate cu dispozitive inteligente ce anunță centrele de colectare corespunzatoare, când sunt necesare colectările.

Utilitatea : proiectul este o aplicaţie ce vine în ajutorul proprietarilor de centre de colectare și sortare a deșeurilor, permiţând desfăşurarea propriei activităţi cu o eficienţă cât mai bună

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

Constrângeri -> key constraint (obs: tabelele apar în ordine alfabetică)

- tabela **ANGAJATI**: - are PRIMARY KEY câmpul ID.
  - are FOREIGN KEY câmpul ID_CENTRU care referă câmpul ID care este PK în tabela CENTRE_DE_COLECTARE.
  - are NOT NULL la câmpurile NUME, TELEFON.

- tabela **CATEGORII_DESEURI:** - are PRIMARY KEY câmpul ID.
  - are FOREIGN KEY câmpul ID_CENTRU care referă câmpul ID care este PK în tabela CENTRE_DE_COLECTARE și câmpul ID_CLIENT care referă câmpul ID care este PK în tabela CLIENTI.
  - are NOT NULL la câmpurile NUME, CANTITATE.

- tabela **CENTRE_DE_COLECTARE:** - are PRIMARY KEY câmpul ID.

- are FOREIGN KEY câmpul ID_ANGAJAT care referă câmpul ID care este PK în tabela ANGAJATI si câmpul ID_VEHICUL care referă câmpul ID care este PK în tabela VEHICULE.
- are NOT NULL la câmpurile CAPACITATE, LOCATIE.

- tabela **CENTRU_CATEGORIE:** - are PRIMARY KEY câmpurile ID_CENTRU, ID_CATEGORIE.

- tabela **CLIENTI:** - are PRIMARY KEY câmpul ID.
- are FOREIGN KEY câmpul ID_CATEGORIE care referă câmpul ID care este PK în tabela CATEGORII_DESEURI.
- are NOT NULL la câmpurile NUME, ADRESA, TELEFON, SPECIALIZARE.

- tabela **CLIENT_CATEGORIE:** - are PRIMARY KEY câmpurile ID_CLIENT, ID_CATEGORIE.

- tabela **COSURI_DE_GUNOI:** - este o subentitate a tabelei FURNIZORI.
- are PRIMARY KEY câmpul ID_COS.
- are FOREIGN KEY câmpul ID_DISPOZITIV care referă câmpul ID care este PK în tabela DISPOZITIVE_INTELIGENTE și câmpul ID_CENTRU care referă câmpul ID care este PK în tabela CENTRE_DE_COLECTARE
- are NOT NULL la câmpurile CAPACITATE, ORAS, NUME si la cele care sunt foreign key.

- tabela **DISPOZITIVE_INTELIGENTE:** - are PRIMARY KEY câmpul ID.
- are FOREIGN KEY câmpul ID_CENTRU care referă câmpul ID care este PK în tabela CENTRE_DE_COLECTARE și câmpul ID_COS care referă câmpul ID care este PK în tabela COSURI_DE_GUNOI.

- tabela **FURNIZORI:** - are PRIMARY KEY câmpul ID_FURNIZOR.
- are FOREIGN KEY câmpul ID_CENTRU care referă câmpul ID care este PK în tabela CENTRE_DE_COLECTARE.
- are NOT NULL la câmpul NUME.

- tabela **LOCATII:** - are PRIMARY KEY câmpurile ID_COS, LONGITUDINE, LATITUDINE.
- are NOT NULL la câmpurile ORAS, JUDET.

- tabela **SOFERI:** - este o subentitate a tabelei ANGAJATI.
- are PRIMARY KEY câmpul ID_SOFER.
- are FOREIGN KEY câmpul NUMAR_VEHCUL care referă câmpul ID care este PK în tabela VEHICULE.
- are NOT NULL la câmpul NUME.

- tabela **SOFER_VEHICUL:** - are PRIMARY KEY câmpurile ID_SOFER, NUMAR_VEHICUL.

- tabela **UTILIZATORI:** - are PRIMARY KEY câmpul ID.

- are FOREIGN KEY câmpul ID_CATEGORIE care referă câmpul ID care este PK în tabela CATEGORII_DESEURI.

- tabela **UTILIZATOR_COS:** - are PRIMARY KEY câmpurile ID_UTILIZATOR, ID_COS.

- tabela **VEHICULE:** - are PRIMARY KEY câmpul NUMAR_VEHICUL.
- are FOREIGN KEY câmpul ID_CENTRU care referă câmpul omonim care este PK în tabela CENTRE_DE_COLECTARE și câmpul ID_SOFER care referă câmpul omonim care este PK în tabela SOFERI.

- tabela **VEHICUL_COS:** - are PRIMARY KEY câmpurile ID_COS, NUMAR_VEHICUL.

3. Descrierea entităților, incluzând precizarea cheii primare.

- tabela **ANGAJATI**: - are PRIMARY KEY câmpul ID.
- tabela **CATEGORII_DESEURI:** - are PRIMARY KEY câmpul ID.
- tabela **CENTRE_DE_COLECTARE:** - are PRIMARY KEY câmpul ID.
- tabela **CLIENTI:** - are PRIMARY KEY câmpul ID.
- tabela **COSURI_DE_GUNOI:** - are PRIMARY KEY câmpul ID_COS.
- tabela **DISPOZITIVE_INTELIGENTE:** - are PRIMARY KEY câmpul ID.
- tabela **FURNIZORI:** - are PRIMARY KEY câmpul ID_FURNIZOR.
- tabela **SOFERI: -** are PRIMARY KEY câmpul ID_SOFER.
- tabela **UTILIZATORI:** - are PRIMARY KEY câmpul ID.
- tabela **VEHICULE:** - are PRIMARY KEY câmpul NUMAR_VEHICUL.

4. Descrierea relațiilor, incluzând precizarea cardinalității acestora.

Fiecare utilizator poate folosi unul sau mai multe cosuri de gunoi.

Fiecare cos de gunoi poate fi folosit de unul sau mai multi utilizatori.

Fiecare dispozitiv inteligent poate sa se afle la un singur cos de gunoi.

Fiecare cos de gunoi trebuie sa contina un singur dispozitiv inteligent.

Fiecare dispozitiv inteligent trebuie sa fie monitorizat de un centru de colectare.

Fiecare centru de colectare trebuie sa monitorizeze unul sau mai multe dispozitive inteligente.

Fiecare sofer poate fi angajat de un singur centru de colectare.

Fiecare centru de colectare trebuie sa angajeze unul sau mai multi soferi.

Fiecare sofer poate sa conduca unul sau mai multe vehicule (nu vorbesc de acelasi moment de timp).

Fiecare vehicul poate fi condus de unul sau mai multi soferi (aceeasi observatie).

Fiecare vehicul trebuie sa fie detinut de un singur centru de colectare.

Fiecare centru de colectare trebuie sa detina unul sau mai multe vehicule.

Fiecare cos de gunoi poate fi colectat de unul sau mai multe vehicule (nu e mandatory, pentru ca pot exista cosuri care sa nu se umple niciodata daca nu sunt accesate de multi utilizator => nu vor trimite nicio alarma catre autoritati).

Fiecare vehicul poate sa se deplaseze la unul sau mai multe cosuri de gunoi.

Fiecare furnizor poate furniza un singur centru de colectare.

Fiecare centru de colectare trebuie sa fie furnizat de unul sau mai multi furnizori.

Fiecare centru de colectare trebuie sa contina una sau mai multe categorii de deseuri.

Fiecare categorie de deseuri poate sa fie la unul sau mai multe centre de colectare.

Fiecare categorie de deseuri trebuie sa fie cumparata de unul sau mai multi clienti.

Fiecare client trebuie sa cumpere una sau mai multe categorii de deseuri.

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

- tabela **ANGAJATI**: are 4 atribute de tip number(4): pe cele care se refera la id-uri (PK/FK) si 2 de tip varchar() pentru nume si adresa.

- tabela **CATEGORII_DESEURI:** are 2 atribute de tip number(4): id-ul si cantitatea de deseuri, 2 de tip varchar() pentru nume si culoare, 1 de tip date pentru data_livrarii deseurilor.

- tabela **CENTRE_DE_COLECTARE:** are 4 atribute de tip number(4): pe cele care se refera la id-uri (PK/FK) si pe cel pentru capacitatea maxima de deseuri acceptata de centru, 1 de tip varchar() pentru locatie.

- tabela **CENTRU_CATEGORIE:** are 3 atribute de tip number(4): pe cele care se refera la id-uri (PK/FK) si la numarul echipamentelor specifice unei categorii, cu care e dotat fiecare centru.

- tabela **CLIENTI:** are 2 atribute de tip number(4): id-ul si numarul de telefon, 3 de tip varchar() pentru nume, specializarea de deseuri si adresa

- tabela **CLIENT_CATEGORIE:** are 2 atribute de tip number(4): id-urile, 1 de tip varchar() pentru observatii.

- tabela **COSURI_DE_GUNOI:** are 4 atribute de tip number(4): pe cele care se refera la id-uri (PK/FK), 2 de tip varchar() pentru nume si oras, 1 de tip date pentru data_instalarii.

- tabela **DISPOZITIVE_INTELIGENTE:** are 3 atribute de tip number(4): pe cele care se refera la id-uri (PK/FK), 1 de tip varchar() pentru locatie.

- tabela **FURNIZORI:** are 2 atribute de tip number(4): pe cele care se refera la id-uri (PK/FK), 1 de tip varchar() pentru nume.

- tabela **LOCATII:** are 1 atribut de tip number(4): id-ul, 4 de tip varchar() pentru longitudine, latitudine, oras si judet.

- tabela **SOFERI:** are 4 atribute de tip number(4): pe cele care se refera la id-uri (PK/FK), salariul si telefonul, 3 de tip varchar() pentru nume, tip_permis si adresa.

- tabela **SOFER_VEHICUL:** are 1 atribut de tip number(4): id-ul si 2 de tip varchar() pentru numar_vehicul si observatii

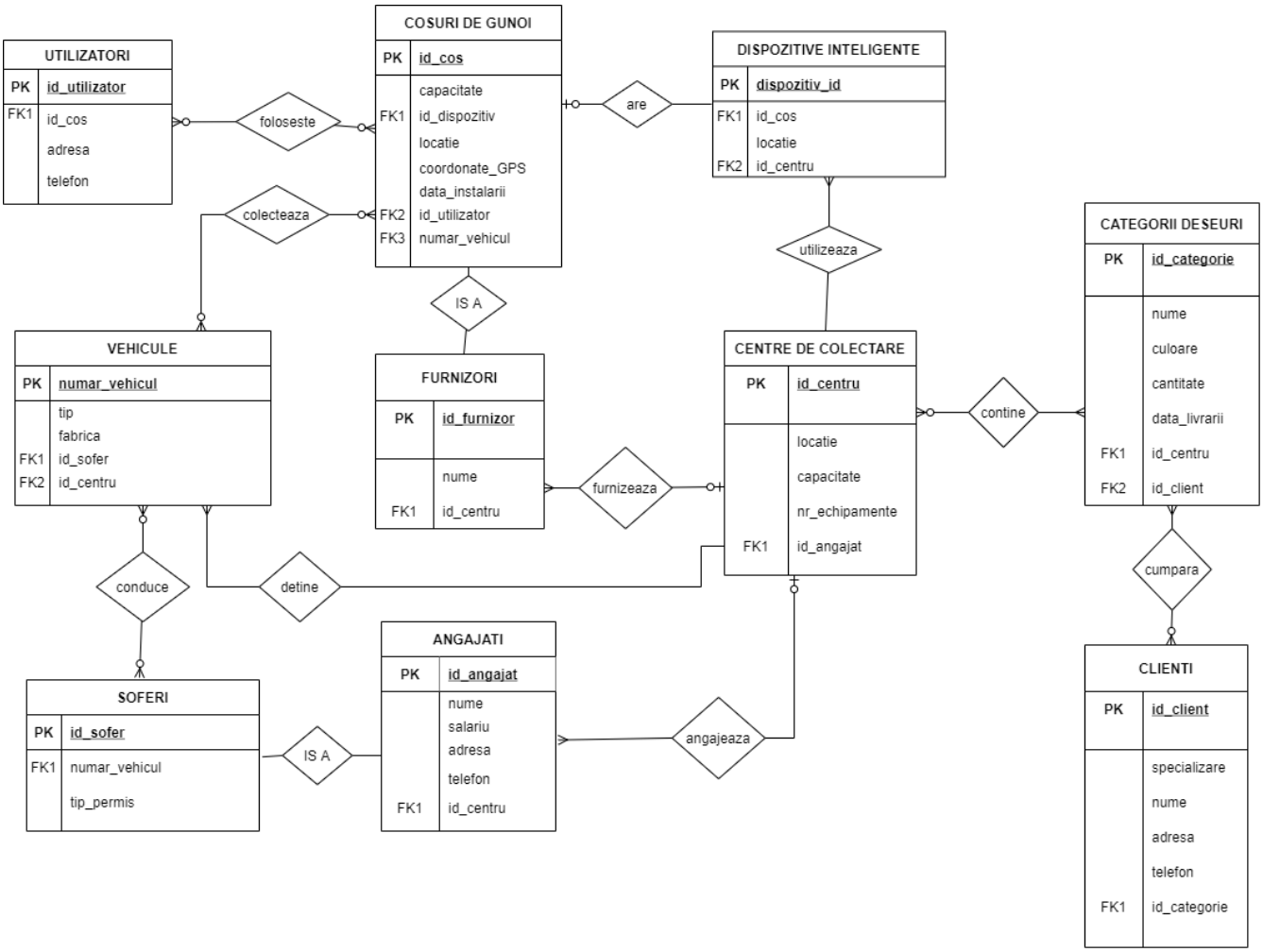
- tabela **UTILIZATORI:** are 2 atribute de tip number(4): id-ul si telefonul, 1 de tip varchar() pentru adresa.

- tabela **UTILIZATOR_COS:** are 2 atribute de tip number(4): id-urile, 1 de tip varchar() pentru observatii.

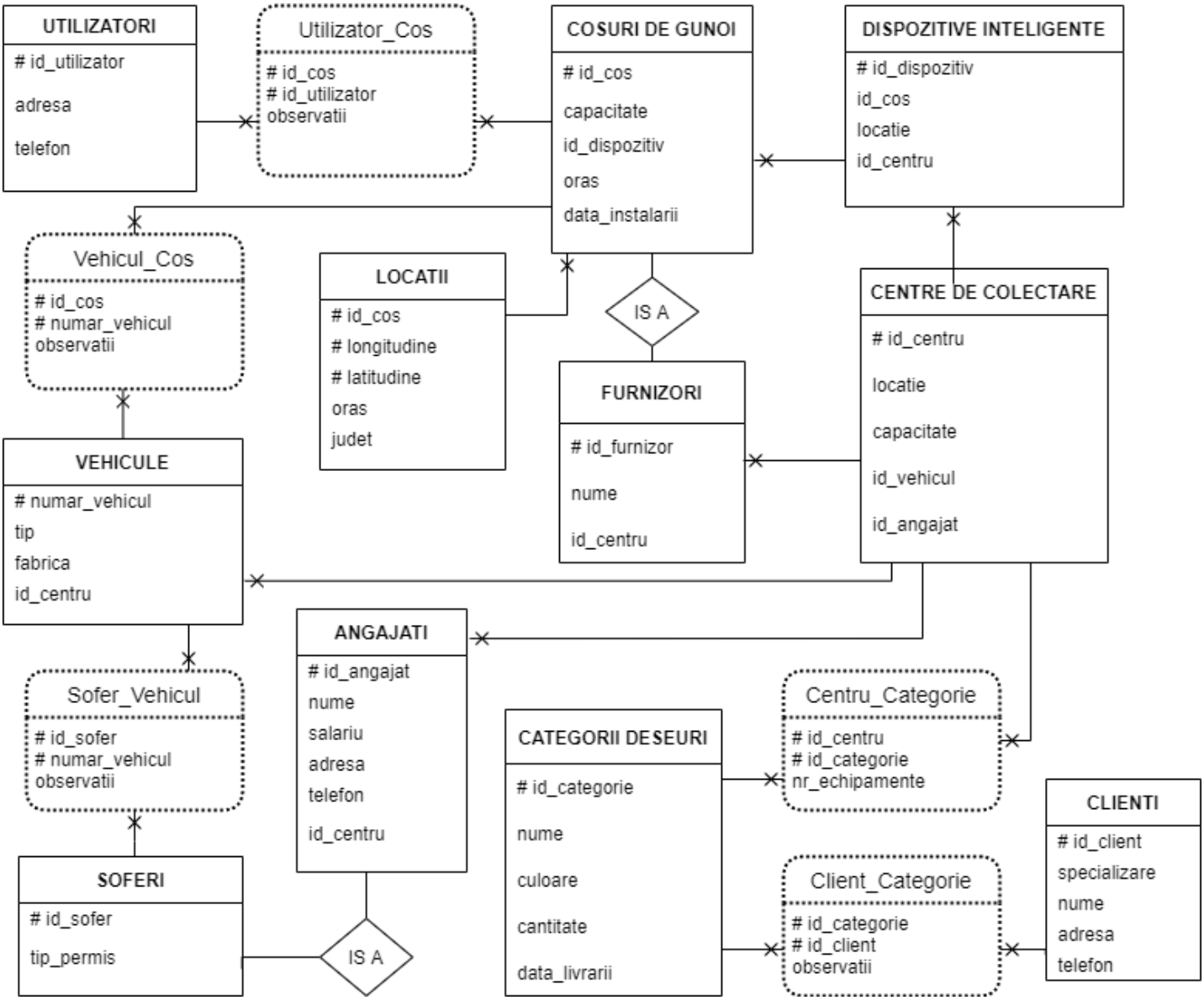
- tabela **VEHICULE:** are 1 atribut de tip number(4): id-ul, 3 de tip varchar() pentru numar_vehicul, tip, fabrica.

- tabela **VEHICUL_COS:** are 1 atribut de tip number(4): id-ul si 2 de tip varchar() pentru numar_vehicul si observatii

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. Scheme relationale:

Centre de colectare (#id, locatie, capacitate, nr_angajati, id_angajat, echipamente)

Categorii deseuri (#id, nume, culoare, data_livrarii, cantitate)

Centru_Categorie (#id_centru, #id_categorie, observatii)

Furnizori (#id, nume, telefon, capacitatea_maxima)

Cosuri de gunoi (#id_cos, capacitate, locatie, data_instalarii, id_dispozitiv)

Utilizator_Cos (#id_cos, #id_utilizator, observatii)

Clienti (#id, specializare, nume, adresa, telefon)

Client_Categorie (#id_client, #id_categorie, observatii)

Angajati (#id, nume, salariu, adresa, telefon, id_centru)

Soferi (#id_sofer, tip_permis)

Dispozitive inteligente (#id_dispozitiv, locatie, id_cos , id_centru)

Utilizatori (#id_utilizator, adresa, telefon)

Vehicule (#numar_vehicul, tip, fabrica, id_centru)

Vehicul_Cos (#id_cos, #numar_vehicul, observatii)

Locatii (#id_cos, #longitudine, #latitudine, oras, judet)

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

Non-FN1: atributul “locatie” (din tabelul COSURI_DE_GUNOI)

COSURI_DE_GUNOI (#id_cos, locatie, etc.)

➔ Devine: COSURI_DE_GUNOI (#id_cos, oras, judet, strada, numar, etc.)

Non-FN2: in tabela UTILIZATOR-COS am atributul “telefon_utilizator” care depinde de id_utilizator dar nu si de id_cos => elimin acest atribut de aici si il las doar in tabela Utilizator

UTILIZATOR-COS (#id_cos, #id_utilizator, telefon_utilizator , observatii)

UTILIZATORI (#id_utilizator, adresa)

➔ Devine: UTILIZATOR-COS (#id_cos, #id_utilizator, observatii)
UTILIZATORI (#id_utilizator, adresa, telefon_utilizator)

Non-FN3: am dependenta tranzitiva intre atributele ce indica locatia (*oras*, *judet*, *strada*) si atributul *coordonate_GPS* din tabela COSURI_DE_GUNOI: id_cos -> locatie -> coordonate_GPS => creez o noua tabela numita LOCATII, cu cheia primara formata din *id_cos* si *oras*, ce contine atributele *longitudine* si *latitudine* (in loc de *coordonate_GPS)* si pe cele ce indica locatia, iar in COSURI_DE_GUNOI pastrez doar *id_cos* si *oras*

COSURI_DE_GUNOI (#id_cos, coordonate-GPS, etc.)

➔ Devine: COSURI_DE_GUNOI (#id_cos, oras, etc.)
LOCATII (#id_cos, #longitudine, #latitudine, oras, judet)

10. Crearea tabelelor în SQL și inserarea de date coerente în fiecare dintre acestea (minimum 5 înregistrări în fiecare tabel neasociativ; minimum 10 înregistrări în tabelele asociative=provin din relatii many-to-many).
➔ Zavoiu_Andreea_creare_inserare.sql

11. Formulați în limbaj natural și implementați 5 cereri SQL complexe ce vor utiliza, în ansamblul lor, următoarele elemente:

• operație join pe cel puțin 4 tabele
• filtrare la nivel de linii -> where
• subcereri sincronizate în care intervin cel puțin 3 tabele • subcereri nesincronizate în care intervin cel puțin 3 tabele -> subcerere ce poate rula independent de cererea principala
• grupări de date (-> max), funcții grup (-> group by), filtrare la nivel de grupuri -> having
• ordonări -> order by
• utilizarea a cel puțin 2 funcții pe șiruri de caractere (-> lower si upper), 2 funcții pe date calendaristice (-> trunc si to_char), a funcțiilor NVL și DECODE, a cel puțin unei expresii CASE
• utilizarea a cel puțin 1 bloc de cerere (clauza WITH)

➔ Zavoiu_Andreea_exemple.sql

12. Implementarea a 3 operații de actualizare sau suprimare a datelor utilizând subcereri.

```
59          -- Cerinta 12
60  -- Stergerea locatiilor din judetele care incep cu litera a
61  delete from locatii
62  where lower(judet) like 'a%';
63
64  -- Marirea salariului tuturor angajatilor din tabelul ANGAJATI cu 5%:
65  update angajati
66  set salary = salary * 0.05;
67  commit;
68
69  -- Sa se stearga datele din tabelul SOFERI.
70  delete from soferi;
71
```

13. Crearea unei secvențe ce va fi utilizată în inserarea înregistrărilor în tabele

```
244
245  ------------------------------------------------------------------------------------------------
246  CREATE TABLE utilizatori(
247                          id_utilizator number(4) PRIMARY KEY,
248                          telefon number(10),
249                          adresa varchar2(100));
250
251  CREATE SEQUENCE s INCREMENT BY 1 START WITH 1;
252  INSERT INTO utilizatori (id_utilizator, telefon, adresa) VALUES (s.nextval, 0733195500, 'Bacau
253  INSERT INTO utilizatori (id_utilizator, telefon, adresa) VALUES (s.nextval, 0788499350, 'Baico
254  INSERT INTO utilizatori (id_utilizator, telefon, adresa) VALUES (s.nextval, 0701239973, 'Sibiu
255  INSERT INTO utilizatori (id_utilizator, telefon, adresa) VALUES (s.nextval, 0722451306, 'Bucur
256  INSERT INTO utilizatori (id_utilizator, telefon, adresa) VALUES (s.nextval, 0714136909, 'Craio
257
258  ------------------------------------------------------------------------------------------------
259  CREATE TABLE utilizator_cos(
260                              id_cos number(4) NOT NULL,
261                              id utilizator number(4) NOT NULL,
```

Script Output ×

Task completed in 0.776 seconds

```
Sequence S created.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.
```

14. Crearea unei vizualizări compuse. Dați un exemplu de operație LMD permisă pe vizualizarea respectivă și unul de operație LMD nepermisă.

```
74          -- Cerinta 14
75
76  create or replace view employee as
77      (select a.id_angajat, a.id_centru, nume, telefon, capacitate
78      from angajati a
79      join centre_de_colectare cdc on a.id_centru = cdc.id_centru
80      where lower(locatie) = 'bucuresti');
81
82  -- permisa:
83  insert into employee (id_angajat, id_centru, nume, telefon) values (8, 2, 'Mari
84  -- nepermisa:
85  insert into employee (capacitate) values (18);
86  commit;
87
```

Script Output x | Query Result x

Task completed in 0.255 seconds

```
1 row inserted.


Error starting at line : 85 in command -
insert into employee (capacitate) values (18)
Error at Command Line : 85 Column : 23
Error report -
SQL Error: ORA-01779: cannot modify a column which maps to a non key-preserved table
01779. 00000 -  "cannot modify a column which maps to a non key-preserved table"
*Cause:    An attempt was made to insert or update columns of a join view which
           map to a non-key-preserved table.
*Action:   Modify the underlying base tables directly.
```

15. Crearea unui index care să optimizeze o cerere de tip căutare cu 2 criterii. Specificați cererea.

```
        -- Cerinta 15

-- Se cer id-ul si orasul cosurilor de gunoi.
create index cautare_cos on cosuri_de_gunoi (id_cos, oras);
```

16. Formulați în limbaj natural și implementați în SQL: o cerere ce utilizează operația outer join pe minimum 4 tabele și două cereri ce utilizează operația division.

```
-- 3. Sa se clasifice castigurile angajatilor, inclusiv pentru cei care nu sunt soferi si sa se mai
-- afiseze numele, tipul permisului si fabrica vehiculelor pentru toti angajatii si toate vehiculele.
select a.nume,
    case when a.salariu > 3000 then 'high'
         when a.salariu > 2000 then 'average'
         else 'low' end salariu,
    tip_permis "sunt angajatii soferi sau nu?", fabrica "au vehiculele soferi sau nu?"
from angajati a
left outer join soferi s on a.id_angajat = s.id_sofer -- leg angajatii de vehicule prin intermediul soferilor
left outer join sofer_vehicul sv on sv.id_sofer = s.id_sofer
-- vreau sa-mi apara si soferii care nu au vehicul asignat
full outer join vehicule v on v.numar_vehicul = sv.numar_vehicul ;
```

```
        -- Cerinta 16

-- Sa se afiseze id-ul centrelor care nu au inca niciun angajat.
select *
from centre_de_colectare cdc
where not exists (select 1
                  from angajati a
                  where cdc.id_centru = a.id_centru);



-- Sa se afiseze furnizorii al caror nume incepe cu litera c dar nu sunt cosuri de gunoi.
select id_furnizor, nume
from furnizori
where lower(nume) like 'c%'
minus
select id_furnizor, nume
from furnizori
where lower(nume) like '%cos de gunoi%';
```

17. Optimizarea unei cereri, aplicând regulile de optimizare ce derivă din proprietățile operatorilor algebrei relaționale. Cererea va fi exprimată prin expresie algebrică, arbore algebric și limbaj (SQL), atât anterior cât și ulterior optimizării

Cererea in limbaj natural:

-- Sa se afiseze numele angajatilor care au lucrat la prima livrare de deseuri din centrul de colectare din Bucuresti.

Cererea in SQL:

```
select nume
from angajati a
join centre_de_colectare c on a.id_angajat = c.id_angajat
join centru_categorie cc on c.id_centru = cc.id_centru
join categorii_deseuri cd on cc.id_categorie =  cd.id_categorie
where lower(id_centru) = 'bucuresti' and cd.data_livrarii = (
                                    select min(cd2.data_livrarii)
                                    from categorii_deseuri cd2
                                    where cd2.id_categorie = cd.id_categorie);
```

Cererea in algebra relationala:

R1 = PROJECT (ANGAJATI, nume, id_angajat)

S1 = SELECT (CENTRE_DE_COLECTARE, lower(id_centru) = 'bucuresti')

S2 = PROJECT (S1, id_centru)

T1 = PROJECT (CATEGORII_DESEURI, id_categorie, id_centru, data_livrarii)

T2 = SELECT (T1, data_livrarii = min(data_livrarii))

J1 = JOIN (R1, T2)

J2 = JOIN (J1, CENTRU_CATEGORIE)

Rezultat = JOIN (J2, S2)

Cererea ca expresie algebrica:

R1 = Π nume,id_angajat (ANGAJATI)

S1 = σ lower(id_centru) = 'bucuresti' (CENTRE_DE_COLECTARE)

S2 = Π id_centru (S1)

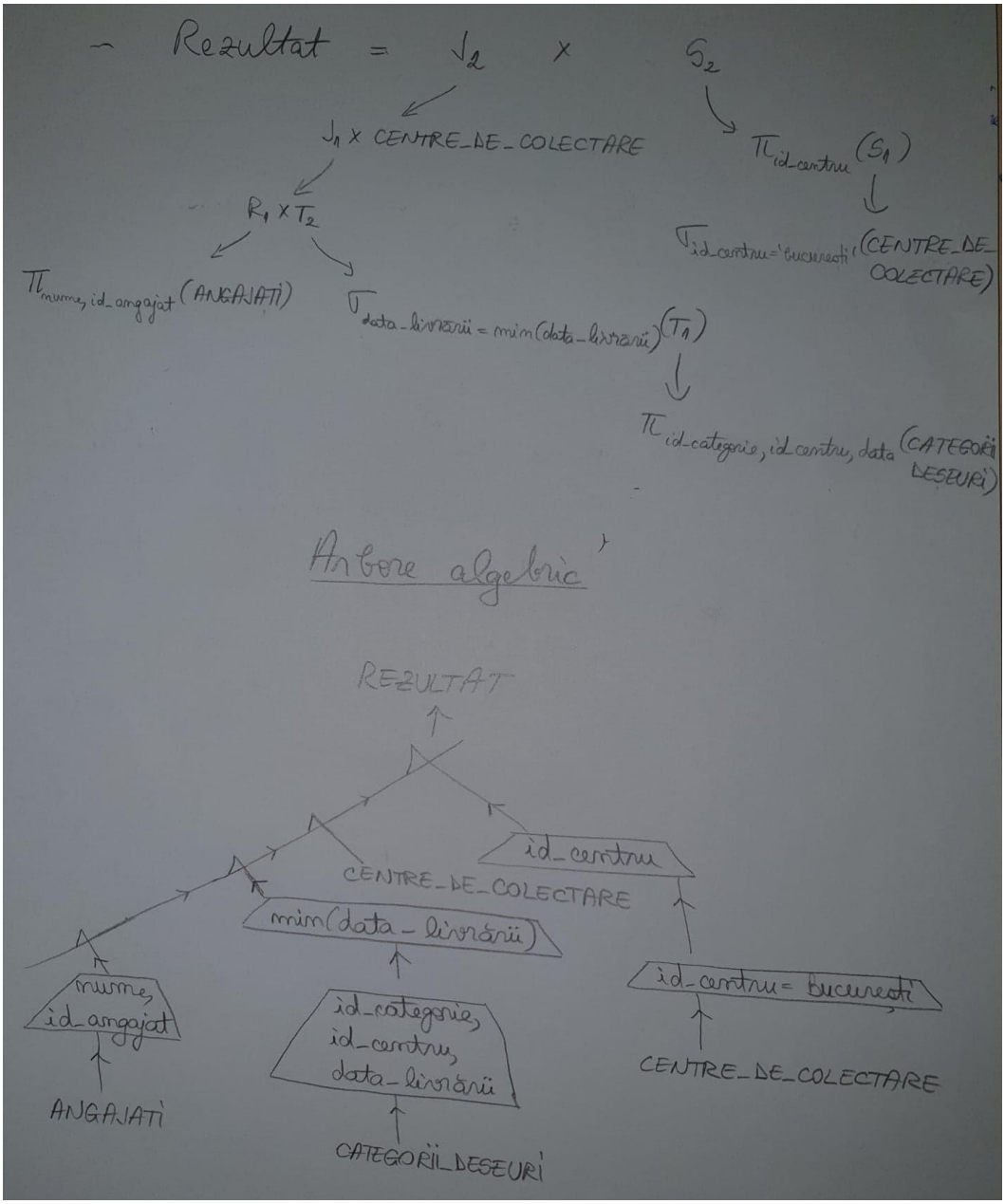
T1 = Π id_categorie, id_centru, data_livrarii (CATEGORII_DESEURI)

T2 = σ data_livrarii = min(data_livrarii) (T1)

J1 = R1 × T2

J2 = J1 × CENTRE_DE_COLECTARE

Rezultat = J2 × S2



18. a. Realizarea normalizării BCNF, FN4, FN5.
b. Aplicarea denormalizării, justificând necesitatea acesteia.

Non-BCNF: in tabelul Sofer :

numar_vehicul -> id_sofer si nume

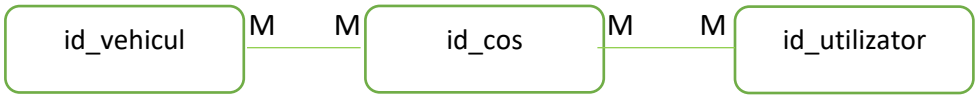
id_sofer -> numar_vehicul

Deci daca cheia primara ar fi fost reprezentata de id_sofer si nume, tabelul ar fi fost in BCNF

Non-FN5 (dependente multivaloare) : in tabelul Cos de gunoi :

id_cos ->-> id_vehicul

id_cos ->-> id_utilizator



Solutie: creez tabele intermediare intre cosuri de gunoi, vehicule si utilizatori care sa contina aceste atribute in pereche.