Exerciții EVA și UML

① Author (name) 1..2 Authorship 1..3 Book (title) 1..5 Readership 0..4 Reader (name)

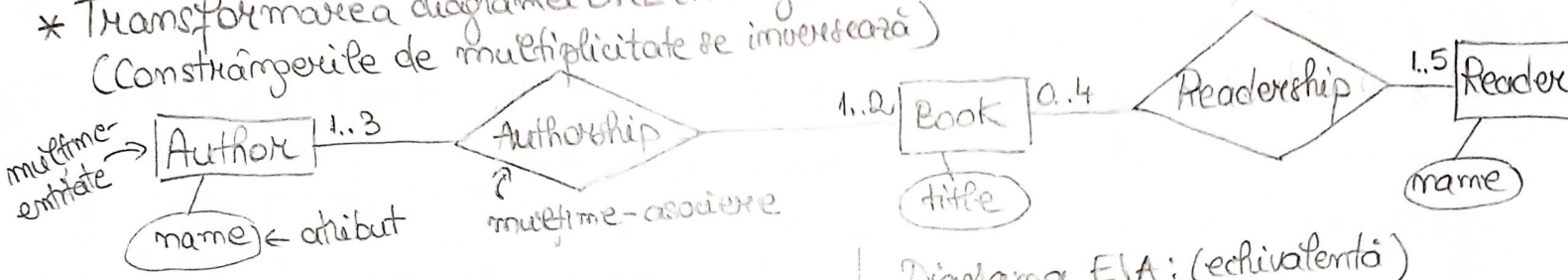
Readership = asociere (binară, fără atribute proprii)
Reader = clasă
name = atribut

a) b)
- Un autor scrie între 1 și 3 cărți.
- O carte este scrisă de 1, maxim 2 autori.
- O carte este citită de între 0 și 4 cititori.
- Un cititor citește între 1 și 5 cărți.

Dc. ∃ 6 autori ⇒ nr. minim de cărți este 3 ⇒ nr. minim de cititori este 0
nr. maxim de cărți este 18 nr. maxim de cititori este 18·4 = 72

Dc. ∃ 6 cititori ⇒ nr. minim de cărți este 2 ⇒ nr. minim de autori este 1
nr. maxim de cărți este ∞ nr. maxim de autori este ∞

* Transformarea diagramei UML în diagramă EVA:
(Constrângerile de multiplicitate se inversează)



② Diagrama UML:

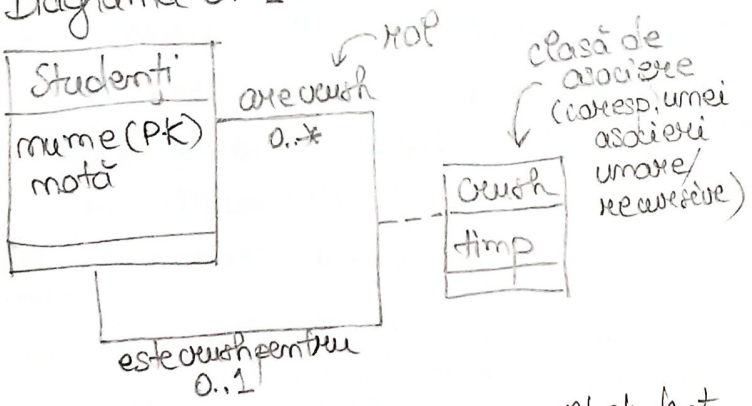
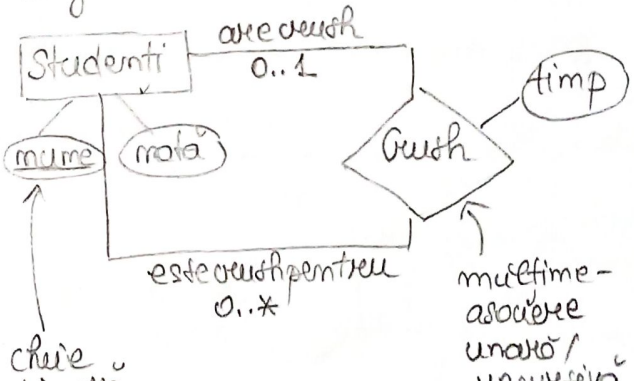


Diagrama EVA: (echivalentă)



Un student are ca și crush maxim 1 alt student.
Un student este crush pt. între 0 și maxim oricâți alți studenți.

Studenți (nume, notă)
Crush (timp, nume1, nume2) — chei străine
nume1 = numele stud care are crushul

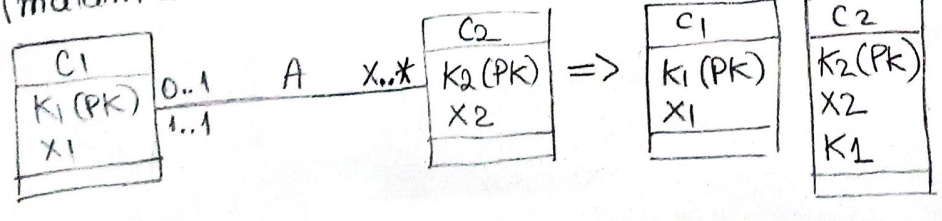
- Calculul de chei pentru asocieri / clase de asociere / mulțimi asociere:

* Chei străine: C1 (K1 (PK), X1) X..Y A U..V C2 (K2 (PK), X2)

C1(K1, X1), C2(K2, X2), A(K1, K2)
K1 și K2 sunt chei străine pt A

(chei primare asociate clase/mulțimilor-entități implicate în asociere / clasa de asociere / mulțimea-asociere)

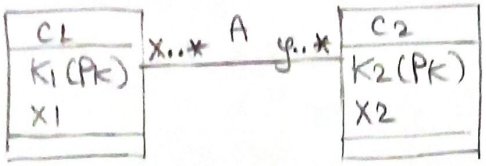
Obs: În cazul asocierilor 1 la mulți / mulți la 1 putem elimina asocierea / clasa de asociere (mutăm atributele acesteia - inclusiv cheile în clasa corresp. lui mulți (*))



C1(K1, X1) C2(K2, X2, K1)
K1 = fostă cheie străină pt A, actuală cheie străină pt C2
(K2 = fostă cheie primară pt A, actuală cheie primară pt C2)

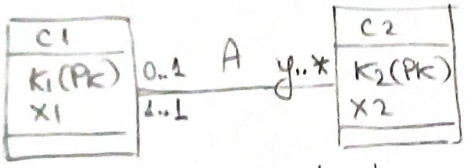
* **Chei primare**

Cazul I: Asociere mulți-mulți ⇒ cheia primară = reuniunea cheilor străine



$C_1(\underline{K_1}, X_1)$
$C_2(\underline{K_2}, X_2)$
$A(\underline{K_1, K_2})$ - cheia primară pt A este (K_1, K_2)

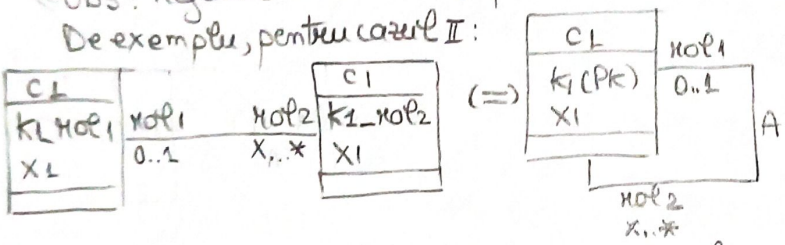
Cazul II: Asociere multi-unul/unul-multi ⇒ cheia primară = cheia străină corresp. clasei de pe ramura multi (*)



$C_1(\underline{K_1}, X_1)$
$C_2(\underline{K_2}, X_2)$
$A(K_1, \underline{K_2})$ - cheia primară pt A este K_2

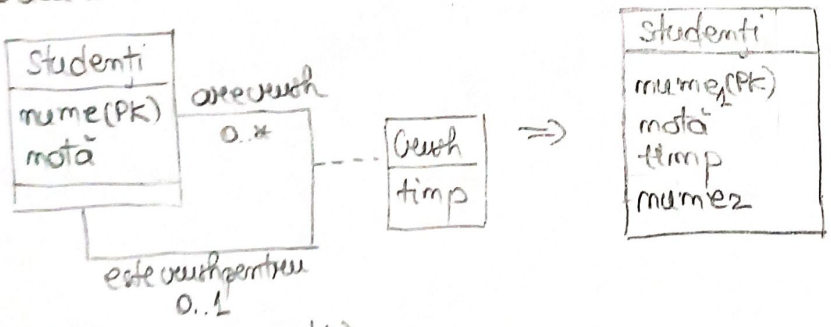
Obs: Regulile se extind și pentru asocieri/clase de asociere unare/recursive

De exemplu, pentru cazul II:



$C_1(\underline{K_1}, X_1)$
$A(K1\_rol1, \underline{K1\_rol2})$
- cheia primară pt A este K1_rol2

Dacă am elimina clasa de asociere Crush:

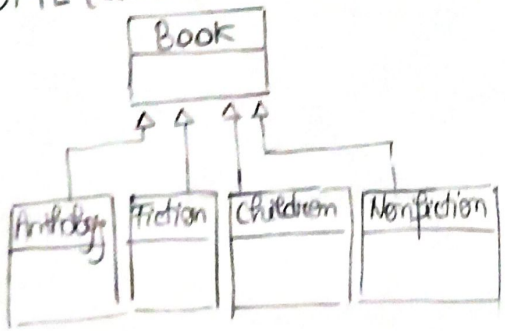


nume1 = numele studentului care are crushul

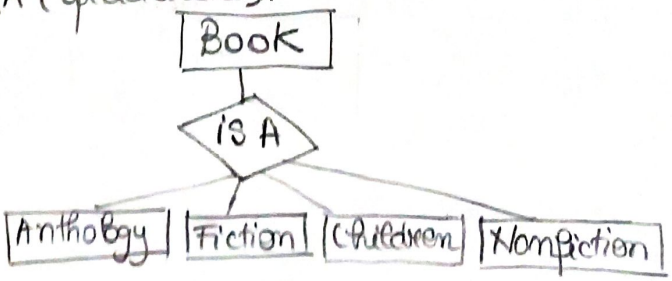
{ Studenti(nume, notă)
Crush(timp, nume1, nume2)

③ Subclasele sunt cu suprapunere, nu disjuncte (De exemplu, cărțile de ficțiune pot fi și cărți pt. copii)
Subclasarea este completă, nu incompletă/parțială (Ficțiune ∪ Nonficțiune = Toate cărțile)

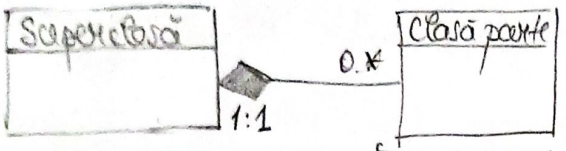
UML (Subclase):



EIA (Specializări):

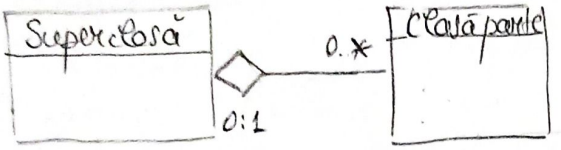


④

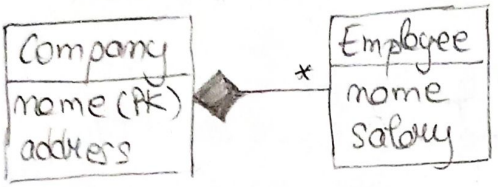


Compoziție Ex: Superclasă: Facultate
Clasă parte: Departament

de obicei entitate slabă (nu are o cheie primară)



Agregare Ex: Superclasă: Facultate
Clasă parte: Birou

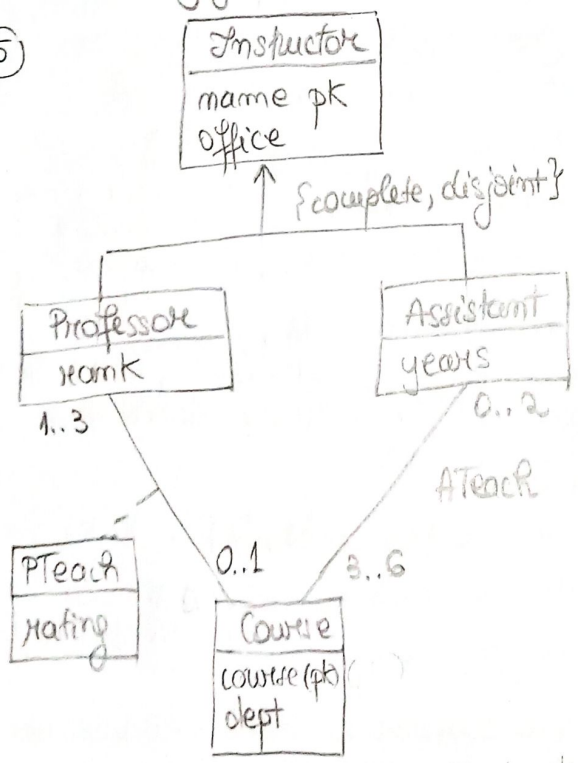


* = 0..*

Company (Cname, address)
Employee (Ename, salary, Cname)
numele coup. este cheie străină pt. clasa Employee

a.) 2 companii nu pot avea același nume DA
b.) 2 angajați nu pot avea același nume NU
c.) 2 companii nu pot avea aceeași adresă NU
d.) 2 angajați nu pot lucra la aceeași adresă NU
e.) Fiecare angajat lucrează pt. cel puțin o companie DA (la agregare ar fi fost NU)
f.) Niciun angajat nu lucrează pt. mai mult de o companie DA
g.) Fiecare companie are cel puțin un angajat NU
h.) 2 angajați cu același nume nu pot lucra pt. aceeași companie NU
i.) 2 angajați cu același nume nu pot lucra pt. companii diferite NU

⑤



a.) Un profesor predă maxim un curs
Un curs este predat de între 1 și 3 profesori.
Un asistent predă între 3 și 6 cursuri.
Un curs este predat de maxim 2 asistenți.

nr. minim de instructori pe curs = 1
1 profesori
0 asistenți

nr maxim de instructori pe curs = 5
3 profesori
2 asistenți

b.) nr. minim de cursuri pt profesori = 0
nr maxim de cursuri pt profesori = 1
nr. minim de cursuri pt. asistenți = 3
nr. maxim de cursuri pt. asistenți = 6

d.) c) Cele 3 maniere de reprezentare a superclasei și subclaselor: (conform slideului 40 din curs)

① Instructor (name, office)
(include toate tuplele
Professor (name, rank)
Assistent (name, years)

② Instructor (name, office) X
(include doar tuplele nespecializate, deci poate fi eliminată subclasarea fiind completă
Professor (name, rank, office)
Assistent (name, years, office)

③ Instructor (name, office, rank, years)

Pentru această situație curentă (subclasare completă, disjunctă), cele mai potrivite variante sunt 2) și 3) (1) repetă informații legate de nume).

Varianta I:

name și course# sunt chei străine pt. PTeach

Apar aici deoarece am eliminat clasa de asociere PTeach

Profesor (name, rank, office, rating, course#)
Asistent (name, years, office)
Course (course#, dept)
ATeach (name, course#) (name și course# sunt chei străine pt. ATeach)

Fiind o asociere mulți-mulți, ambele chei străine formează cheia primară (name, course#).)

Varianta II:

Instructor (name, office, rank, years, rating, course#)
Course (course#, dept)
ATeach (name, course#)

Varianta III (Dacă eliminăm asocierea ATeach, pornind de la varianta I)

Profesor (name, rank, office, rating, course#)
Asistent (name, years, office)
Course (course#, dept, name1, name2)

atributele din clasa ATeach, care nu reprezintă cheia primară din Course, vor fi repetate de 2 ori

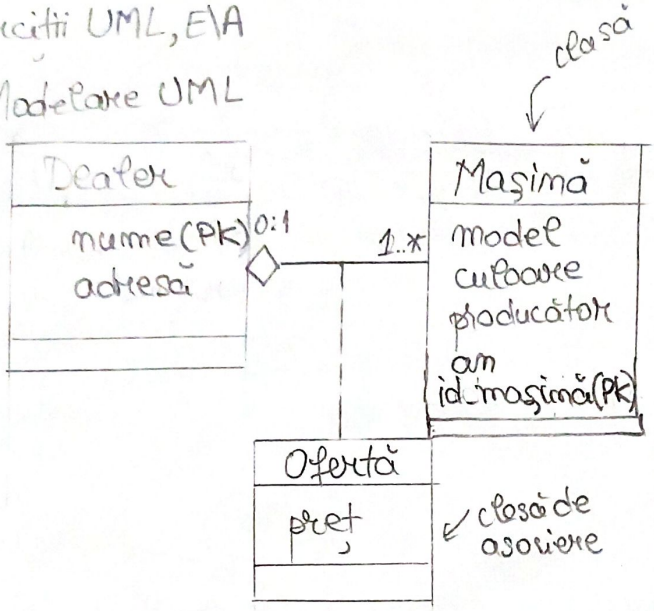
Varianta IV (Dacă pornim de la varianta II și eliminăm ATeach

Instructor (name, office, rank, years, rating, course#)
Course (course#, dept, name1, name2)

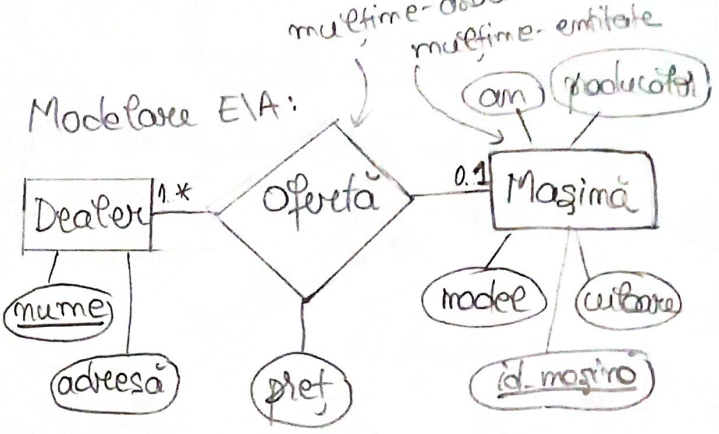
- Având în vedere că un profesor poate să nu predea niciun curs ⇒ rating și course# pot fi NULL, pentru toate cele 4 variante de scheme de mai sus.
- În cazul variantei II și IV fie rank, fie years este NULL pentru fiecare tuplă (deoarece subclasarea este completă și disjunctă).
- În cazul variantelor III și IV, având în vedere că un curs poate fi ținut de între 0 și 2 asistenți ⇒ name1 sau name2 sau amândouă pot fi NULL.

Exerciții UML, E\A

① Modelare UML



Modelare E\A:



Obs1: Deși nu ne menționa explicit, am introdus un identificator unic pt. mașină (id_mașină), pt a putea diferenția mașinile cu același model, an, culoare și producător (vândute de exemplu de același dealer)

Obs2: Clasa Mașină este într-o relație de agregare cu clasa Dealer, astfel clasa parte (Mașină) împrumută cheia primară a clasei mamă (Dealer) ca și cheie străină ⇒ nume este cheie străină pt clasa Mașină

Schema DB: Dealer (nume, adresă)
Mașină (id_mașină, model, culoare, producător, an, nume)
Ofertă (preț, nume, id_mașină)

Obs3: Clasa de asociere Ofertă preia cheile primare ale claselor pe care le conectează ca și chei străine ⇒ nume și id_mașină sunt chei străine pt clasa Ofertă

Obs4: id_mașină este cheie primară pt. clasa Ofertă
(Se aplică regula: cheia primară pt. o asociere/clasă de asociere este cheia primară a clasei asociate mamă a celui multi (*), dacă avem o asociere de tip unul la mulți / mulți la unul - în cazul diagramelor UML)
(Dacă am fi avut o asociere mulți la mulți cheia primară a clasei Ofertă ar fi fost: (nume, id_mașină))

* Dependențe funcționale și multivaluate:

nume → adresă (Numele vânzătorului identifică unic vânzătorul și fiecare vânzător are o singură adresă)

model → producător (2 producători diferiți nu pot avea același model)

nume, an, model → preț (Pe fiecare vânzător, prețul modelului depinde de an, dar nu și de culoare)

producător, model, an ↠ culoare
(sau model, an, producător ↠ vânzător)
(dc. un anumit model dintr-un an există într-o anumită culoare la un vânzător oarecare ⇒ el trebuie să existe în acea culoare la toți vânzătorii care comercializează acel model din acel an)

id_mașină → model, culoare, producător, an

$\Sigma = \{ N \to A,\ M \to P,\ N, Y, M \to D,\ i \to M, C, P, Y \}$

$\Delta = \{ M, A, P \twoheadrightarrow C,\ M, A, P \twoheadrightarrow N \}$

$\{ D(N, A)$
$M(i, M, C, P, Y, N)$
$O(D, N, i)$

D(N,A)
M(i,M,C,P,Y,N)
O(D,N,i)

- Pp. că atributele iau valori atomice ⇒ se respectă 1NF.
- Schema satisface 2NF dc. satisface 1NF și dacă atribut neprim este dependent plin de orice cheie.

$\Sigma = \{N \to A, M \to P, NYM \to D, i \to MCPY\}$

stg.	mijloc	drept.
N	M	A
i	Y	C
		P
		D

$Ni^+ = \{N, A, i, M, C, P, Y, D\} = U \Rightarrow$ Ni este cheie cond.
D, M, A, C, P, Y sunt atr. NEPRIME

∃ dep. $N \to A \Rightarrow$ atributul neprim A nu este dependent plin de cheia Ni ($\exists N \subset Ni$ a.î. avem $N \to A \in \Sigma^+$)

V2) Dacă eram obligați să nu introducem atribute noi, o altă versiune de modelare ar fi putut considerată (model, producător, an, culoare, nume) = cheie primară pt. clasa Mașină. Totodată, fiind vorba de o asociere unele la mulți / mulți la unele, putem eliminăm clasa de asociere oferită.

{ Dealer (nume, adresă)
Mașină (model, culoare, producător, an, nume, plet)

$\Sigma = \{N \to A, M \to P, N, Y, M \to D\}$

D(N, A)
M(M, C, P, A, N, D)

stg.	mijloc	dreapta
N		A
M		P
Y		D

$NMY^+ = U \Rightarrow$ NMY cheie
A, P, D atr. neprime

- La fel ca în versiunea anterioară, schema nu satisface 2NF ⇒ nu satisface nici 3NF, BCNF sau 4NF.

---

② sau PK era (producător, model)

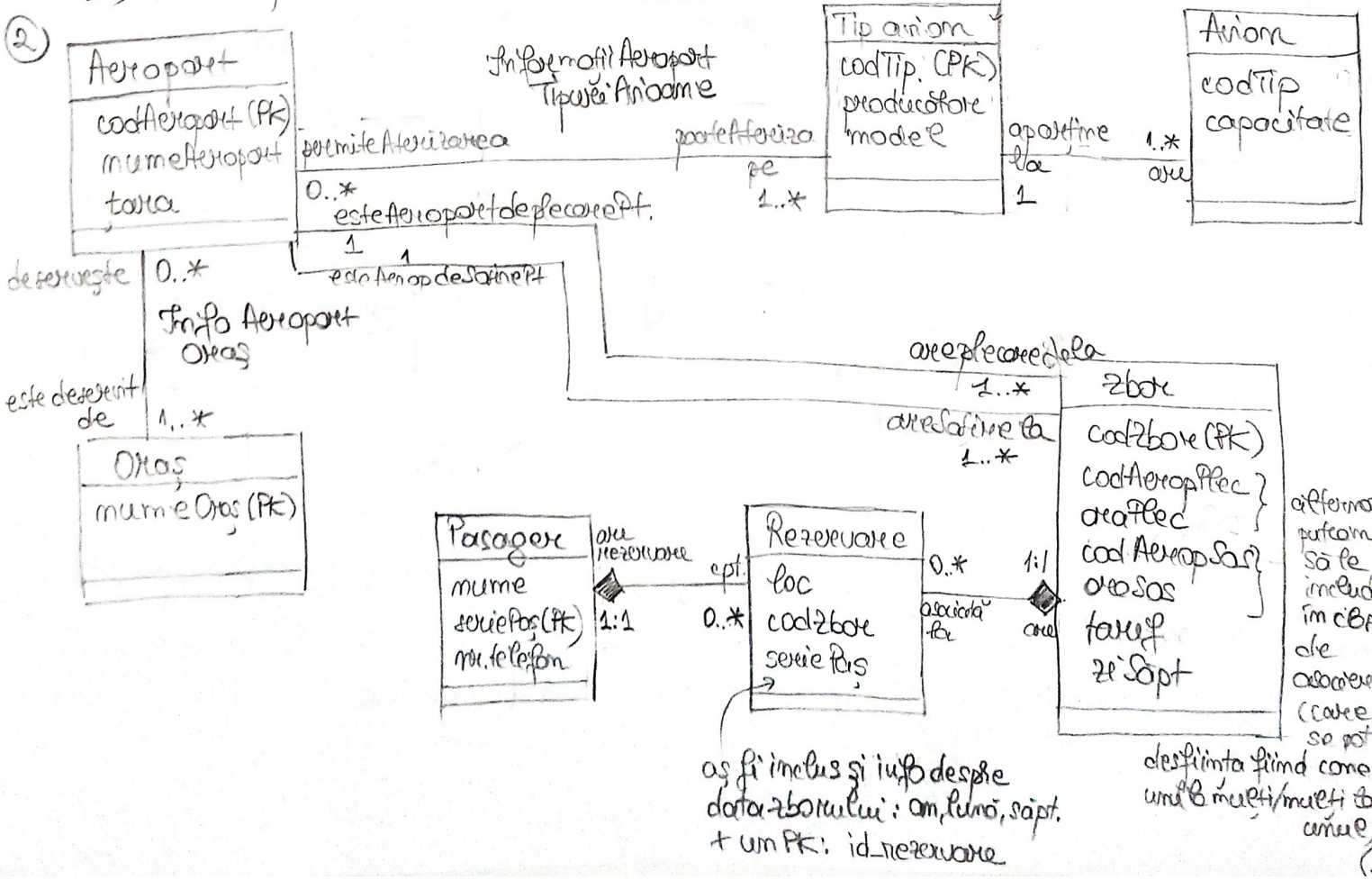
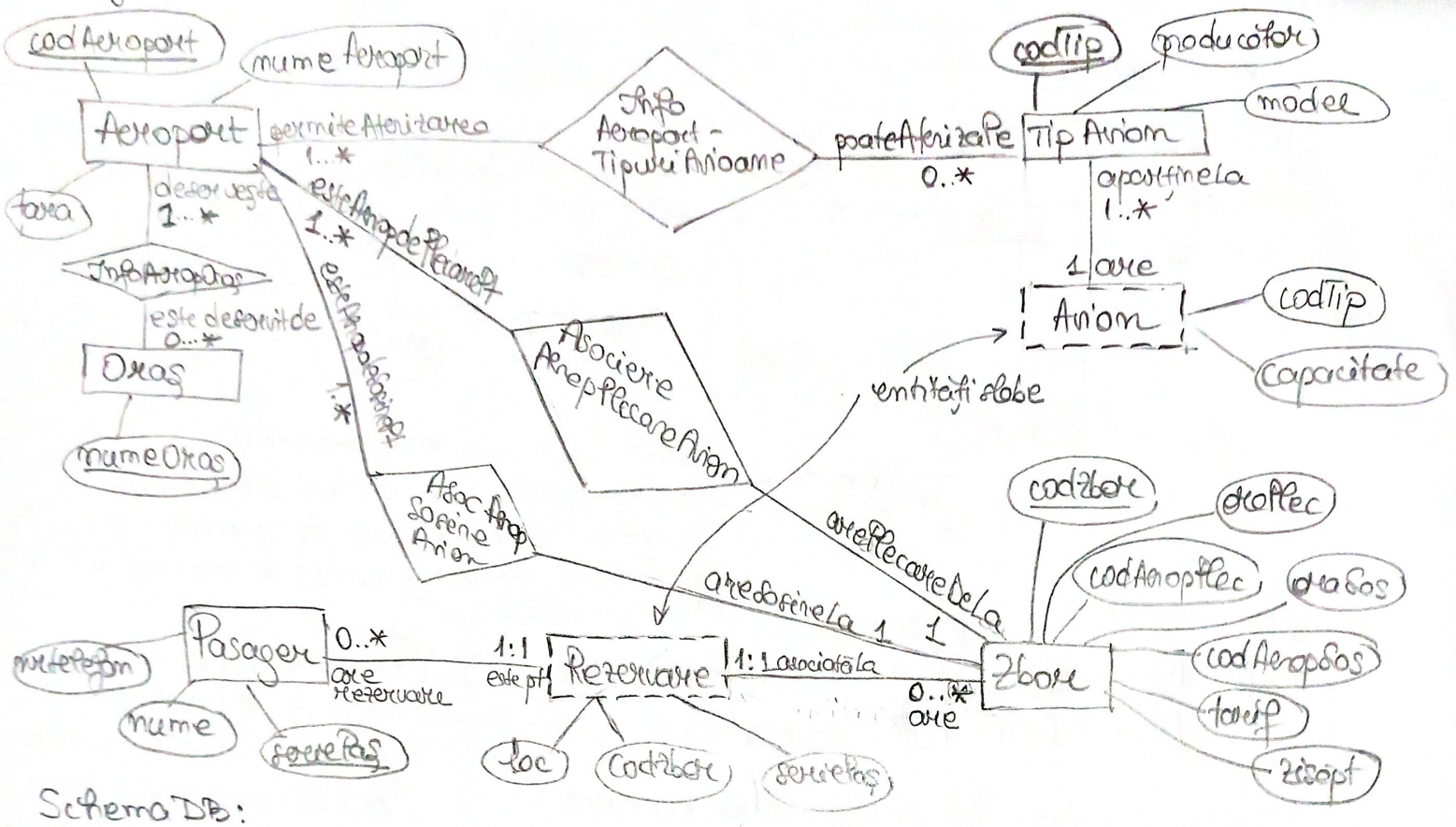


Diagrama E\A asociată:



Schema DB:

Aeroport (codAeroport, numeAeroport, țară)

Oraș (numeOraș)

InfoAeroportOraș (codAeroport, numeOraș) — toate împreună formează cheia primară, fiecare separat sunt chei străine

TipAvion (codTip, producător, model)

InfoAeroportTipuriAvioane (codAeroport, codTip)

Avion (codTip, capacitate) — chei străine

Zbor (codZbor, codAeroPlec, codAeroSosire, dtPlecare, dtSosire, tarif, zisept)

Rezervare (loc, codZbor, serieBuletin) — chei străine

Pasager (nume, serieBuletin, nr. telefon)

Dependențe:

codAeroport → numeAeroport, țară

codTip → producător, model

codZbor → codAeroPlec, codAeroSos, dtPlec, dtSos, tarif, zisept

serieBuletin → nume, nrTelefon

codZbor, loc → serieBuletin (un loc nu poate fi rezervat de 2 persoane)

codZbor, serieBuletin → loc (o persoană rezervă cel mult un loc la un zbor)

∃ o unică cheie: (codAeroport, codTip, codZbor, serieBuletin)

Pp. că schema satisf. 1NF.

∃ dependențe precum codAeroport → numeAeroport, țară, în care atribute neprime (de ex. țară) nu sunt dependente plin de cheie ⇒ schema nu e în 2NF ⇒ nu este nici în 3NF, BCNF sau 4NF.

③ Schema de BD:

Magazine (magId, nume, adresă)
Produse (prodId, nume, descriere)
Vinde (magId, prodId, pret) ((magId, prodId) formează cheie primară => relația dintre Magazine și Produse este de tip multi to multi)
Clienti (clientId, nume, adresa)
Vânzări (clientId, prodId, magId, data)

Diagrama E/A:

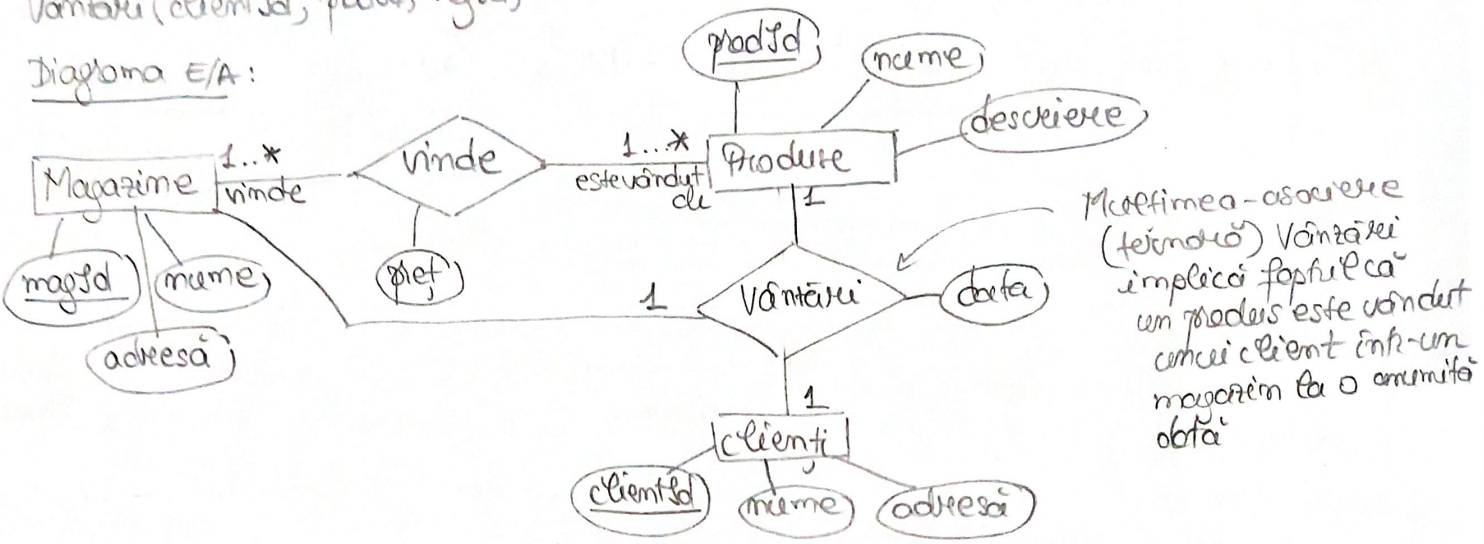
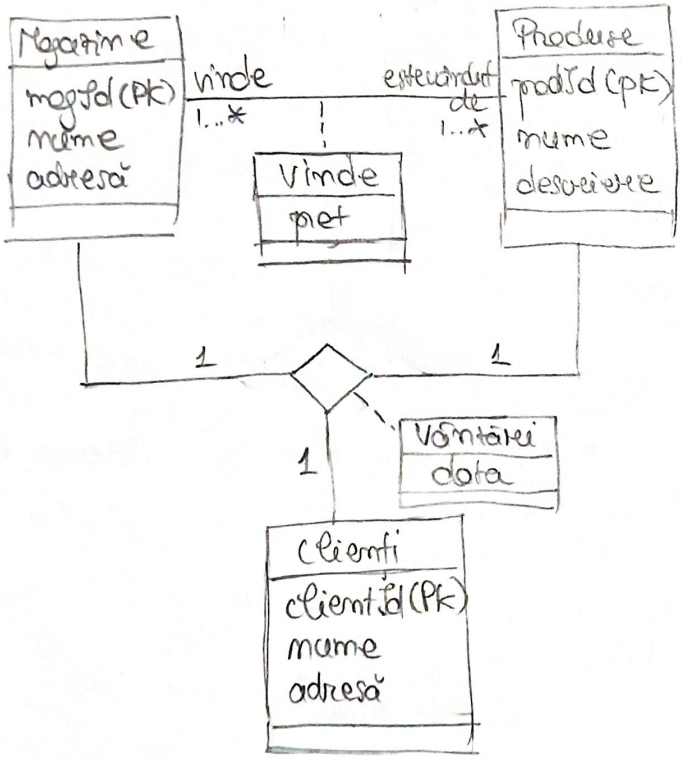


Diagrama UML asociată:



(obs: În general în UML nu se modelează asocierii cu grad > 2 și atributele proprii prin diagrame de clasă)