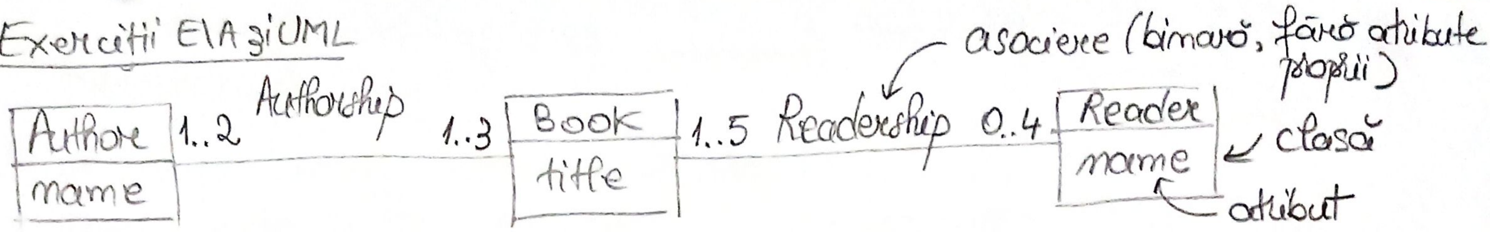
Exerciții E/A și UML

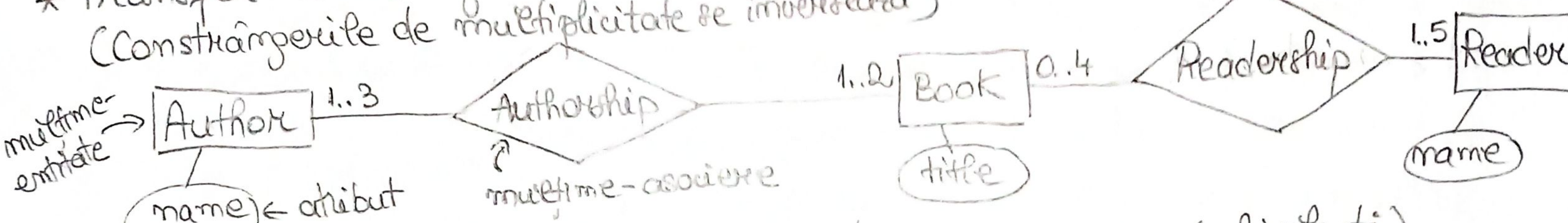
① 

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.

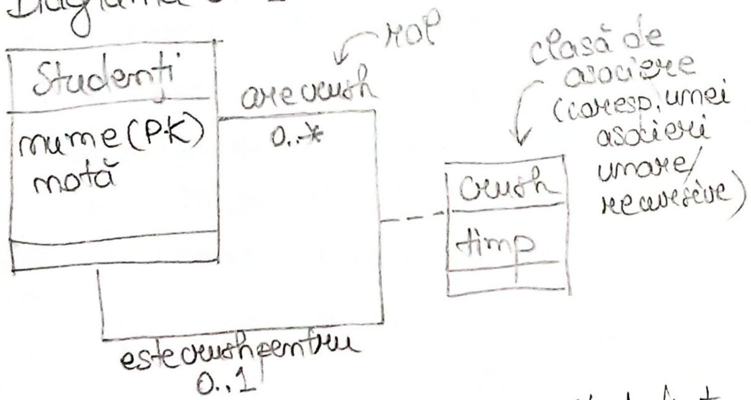
Dc. ∃ 6 autori ⇒ nr. minim de cărți este 6 ⇒ nr. minim de cititori este 0
nr. maxim de cărți este 18 nr. maxim de cititori este $18 \cdot 4 = 72$

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 E/A:
(Constrângerile de multiplicitate se inversează)

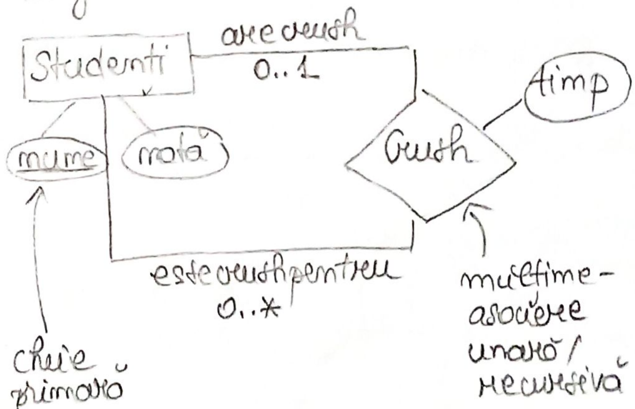


② Diagrama UML:



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

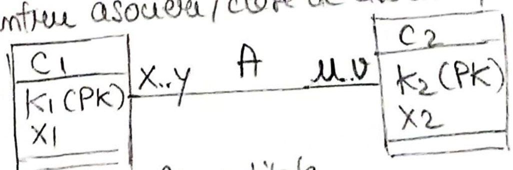
Diagrama E/A: (echivalentă)



Studenți (<u>nume</u>, notă)
Crush (timp, <u>nume1</u>, nume2) chei străine
nume1 = numele stud care are crushul

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

*[Chei străine]:

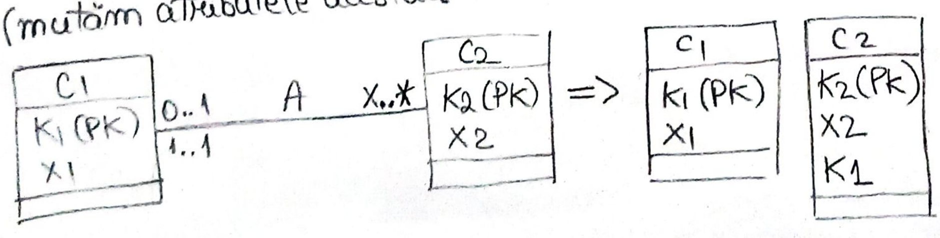


$C_1(\underline{K_1}, X_1)$, $C_2(\underline{K_2}, X_2)$, $A(\underline{K_1, K_2})$

K_1 și K_2 sunt chei străine pt A

||
cheile primare asociate claselor/mulțimilor-entitate 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 coresp. rolului mulți (*))



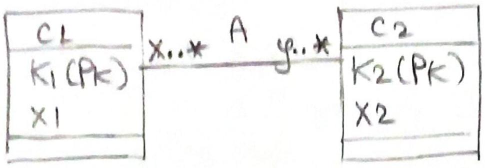
$C_1(\underline{K_1}, X_1)$ $C_2(\underline{K_2}, X_2, K_1)$

K_1 = fostă cheie străină pt A, actuală cheie străină pt C_2

(K_2 = fostă cheie primară pt A, actuală cheie primară pt C_2)

* Chei primare

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

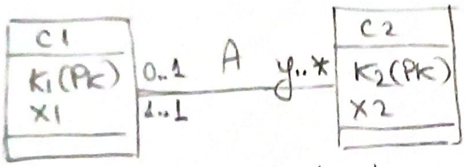


$C_1(\underline{K_1}, X_1)$

$C_2(\underline{K_2}, X_2)$

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

Cazul II: Asociere mulți-unu/unu-mulți => cheia primară = cheia străină coresp. clasei de pe ramura mulți (*)



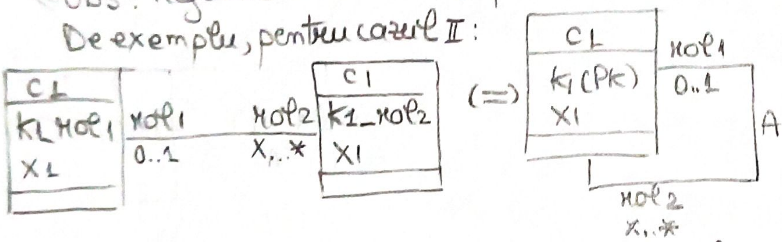
$C_1(\underline{K_1}, X_1)$

$C_2(\underline{K_2}, X_2)$

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

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

De exemplu, pentru cazul II:

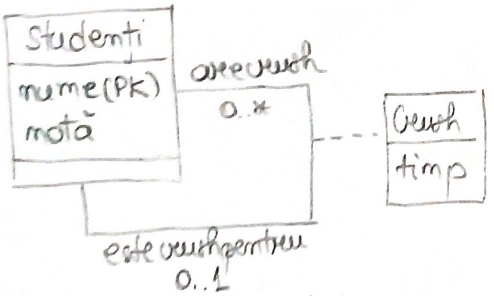


$C_1(\underline{K_1}, X_1)$

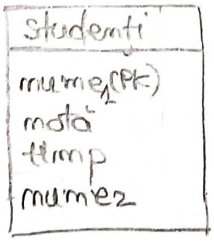
$A(K_1\_rol_1, \underline{K_1\_rol_2})$

- cheia primară pt A este $K_1\_rol_2$

Dacă am elimina clasa de asociere Crush:



=>



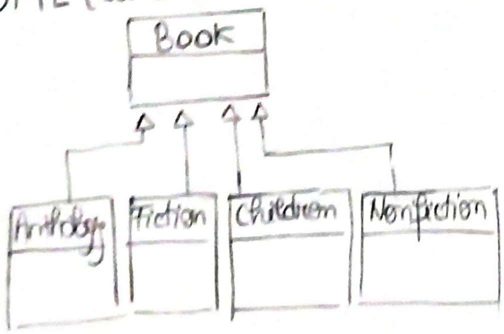
$nume_1$ = numele studentului care are crush-ul

{ Studenti($\underline{nume}$, notă)
Crush(timp, $\underline{nume_1}$, $nume_2$)

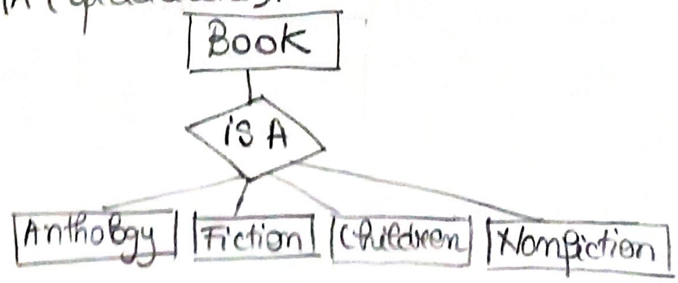
3) Subclasele sunt cu suprapunere, nu disjuncte (De exemplu, cărțile de ficțiune pot fi și cărți pt. copii)

Subclasarea este completă, nu incompletă/parțială (Ficțiune ∪ Nonficțiune = Toate cărțile)

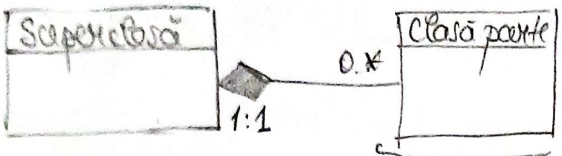
UML (Subclase):



E/A (Specializări):

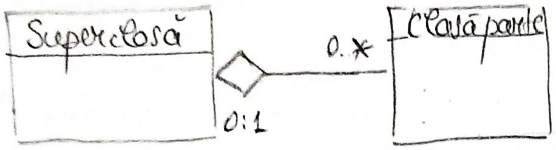


④

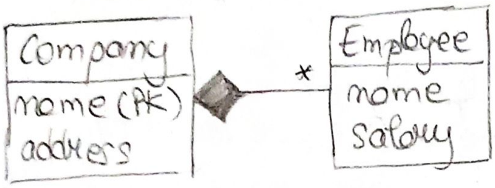


Compoziție Ex: Superclasă: Facultate
Clasă parte: Departament

de obicei entitate slabă (nu are o cheie primară)



Agregare Ex: Superclasă: Facultate
Clasă parte: Birou

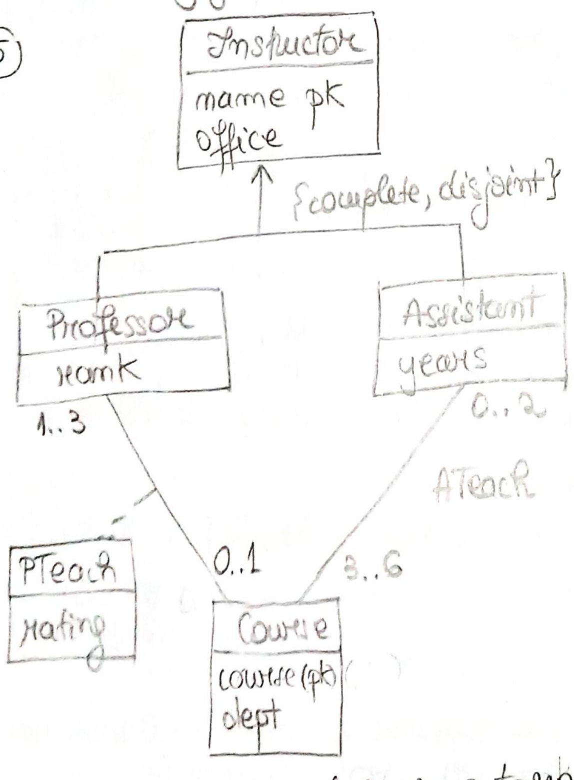


* = 0..*

Company (<u>Cname</u>, address)
Employee (<u>Ename</u>, salary, <u>Cname</u>)
numele comp. 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 nu pot lucra pt. aceeași companie NU
h.) 2 angajați cu același nume nu pot lucra pt. companii diferite NU
i.) 2 angajați cu același nume nu pot lucra

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 = $\underbrace{1}$
1 profesori
0 asistenți

nr. maxim de instructori pe curs = $\underbrace{5}$
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 slide-ului 40 din curs)

① Instructor (<u>name</u>, office)
include toate tuplele
Profesor (<u>name</u>, rank)
Asistent (<u>name</u>, years)

② ~~Instructor (<u>name</u>, office)~~ X
include doar tuple nespecializate, deci poate fi eliminată, subclasarea fiind completă
Profesor (<u>name</u>, rank, office)
Asistent (<u>name</u>, years, office)

③ Instructor (<u>name</u>, office, rank, years)

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

Variant a 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

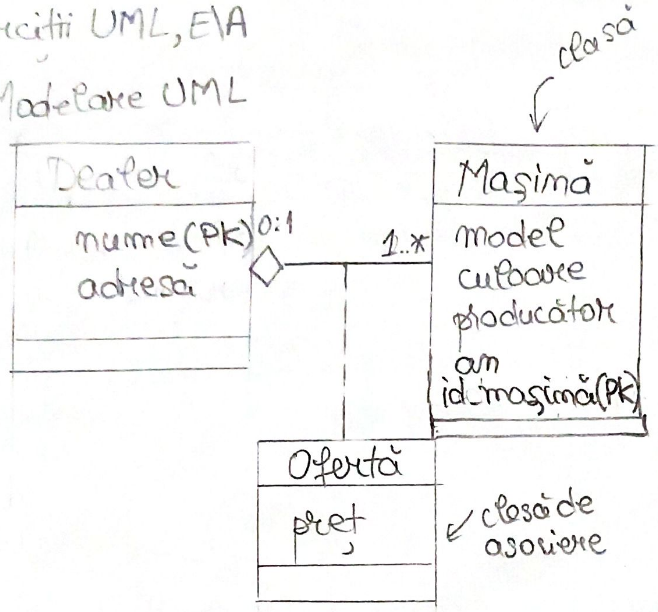
Varianta IV (Dacă pornim de la varianta II și eliminăm ATeach

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

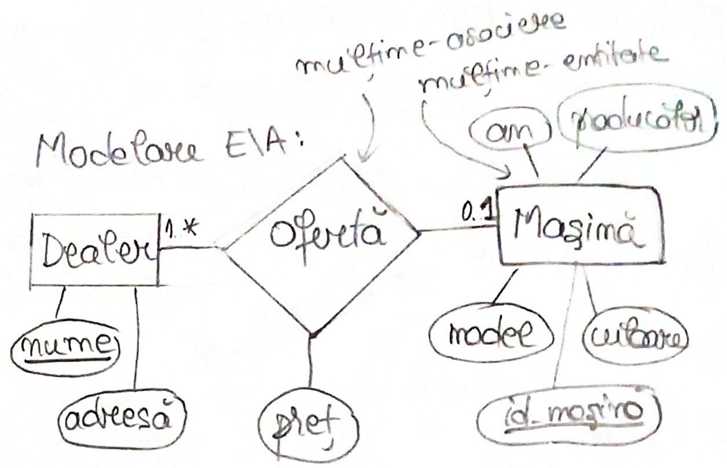
- Având în vedere că un profesor poate să nu predea niciun curs ⇒ rating și course# pot fi NULL, pentru toate cele 4 variante de scheme de mai sus.
- În cazul variantei II și IV fie ranks, fie years este NULL pentru fiecare tuplă (deoarece subclasarea este completă și disjunctă).
- În cazul variantelor 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.

Exercitii UML, E\A

(1) Modelare UML



Modelare E\A:



Obs1: Deși nu se 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)

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

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

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 numărului mulți (*), dacă avem o asociere de tip unu la mulți/mulți la unu - 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 (la fiecare vânzător, prețul modelului depinde de an, dar nu și de culoare)

producător, model, an ↠ culoare
(sau model, an ↠ vânzător)
producător,

(d.c. 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

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

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

$\begin{cases} D(N, A) \\ M(i, M, C, P, Y, N) \\ O(D, N, i) \end{cases}$

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

- Pp. că atributele iau valori atomice $\Rightarrow$ se respectă 1NF.
- Schema satisface 2NF dc. satisface 1NF și orice 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 cand. ; D, M, A, C, P, Y sunt atr. NEPRIME

$\exists$ 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 considera (model, producător, an, culoare, nume) = cheie primară pt. clasa Mașină. Totodată, fiind vorba de o asociere unu la mulți / mulți la unu putem elimina clasa de asociere Ofertă.

Dealer (nume, adresă)
Mașină (model, culoare, producător, an, nume, preț)

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

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

- Ca în versiunea anterioară, schema nu satisface 2NF $\Rightarrow$ nu satisface nici 3NF, BCNF sau 4NF.

(2)

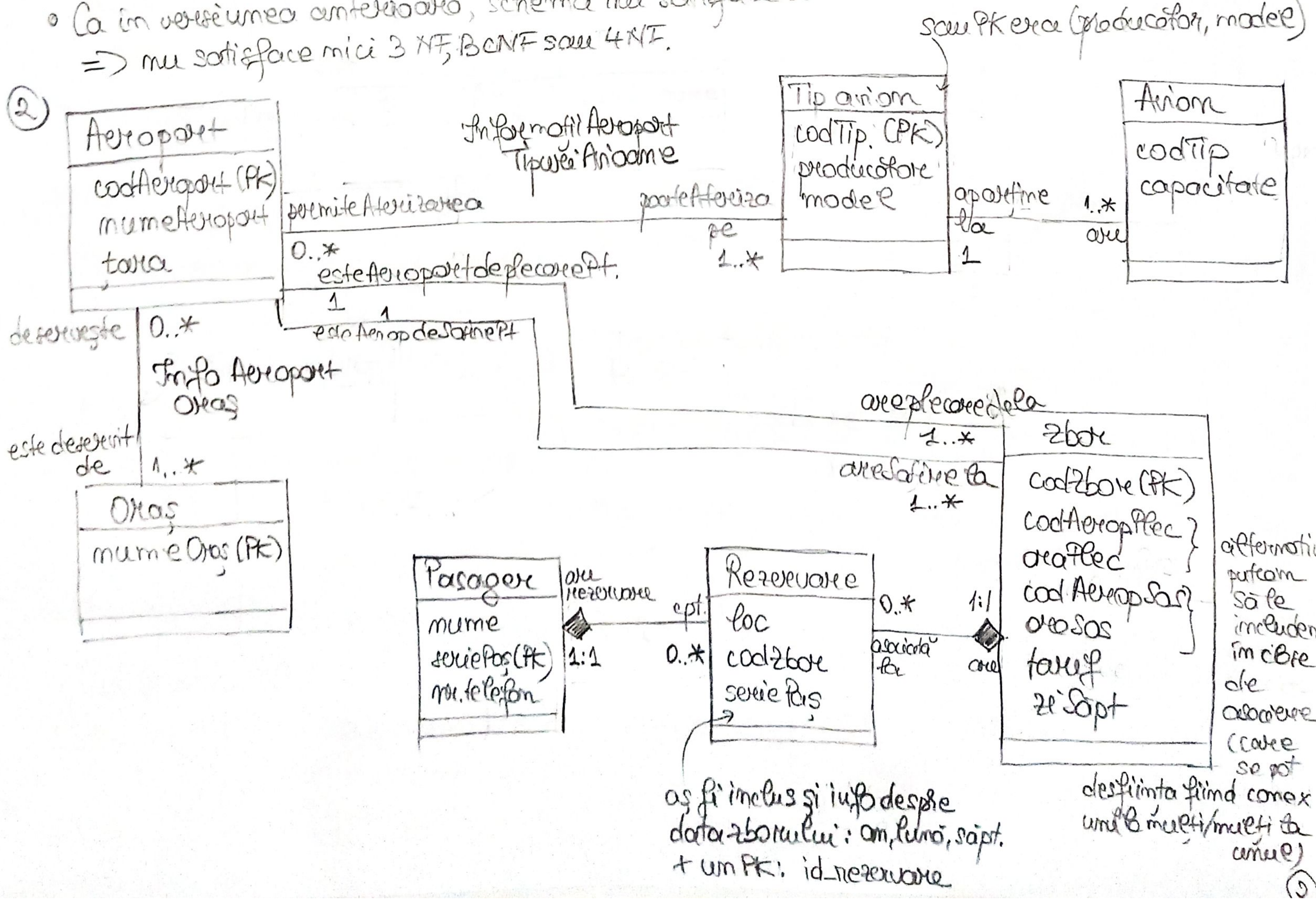
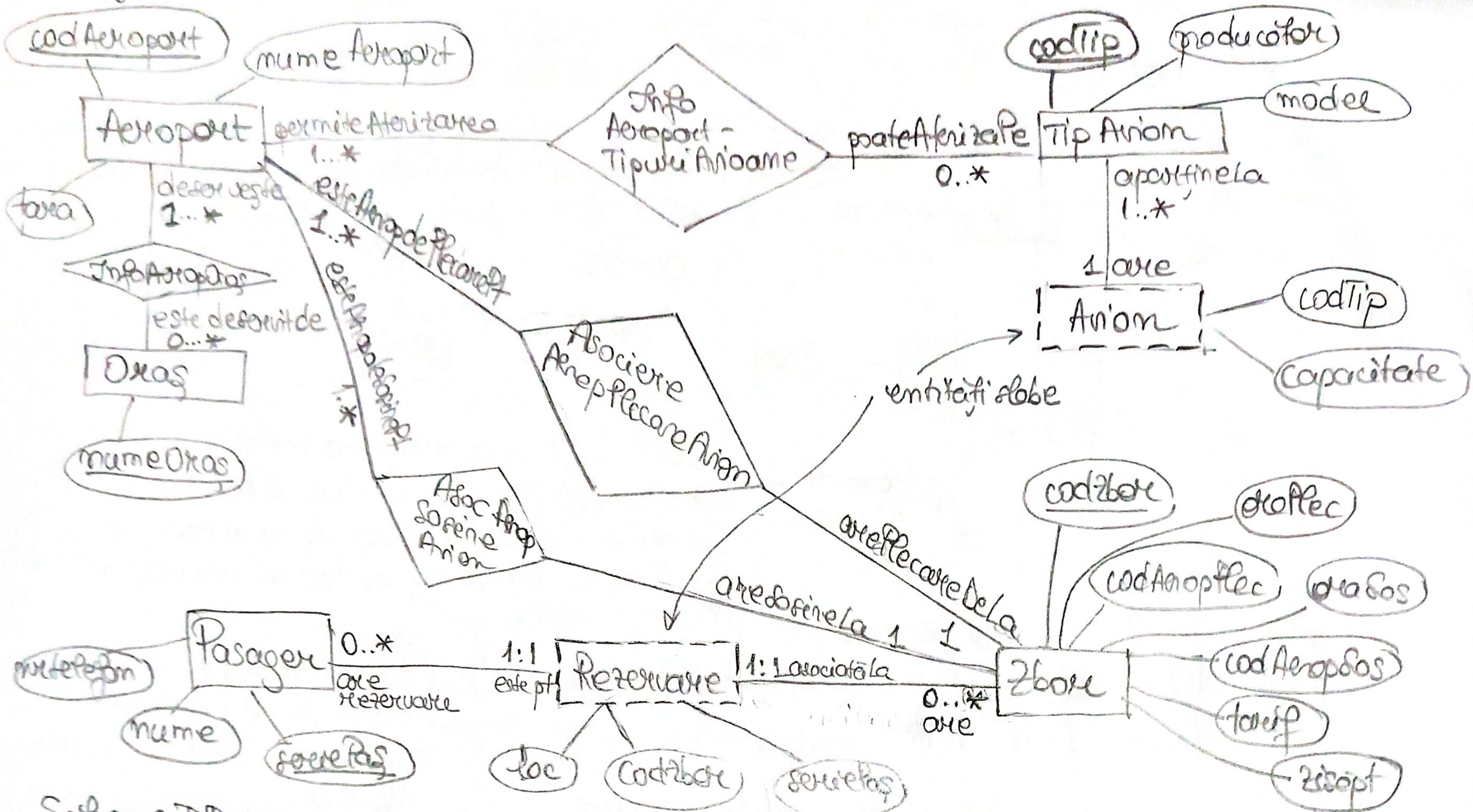


Diagrama E/A asociată:



Schema DB:

- Aeroport (codAeroport, numeAeroport, țară)
- Oraș (numeOraș)
- InfoAeroportOraș (codAeroport, numeOraș) — luate împreună formează cheia primară, luate separat sunt chei străine
- TipAvion (codTip, producător, model)
- InfoAeroportTipuriAvioane (codAeroport, codTip)
- Avion (codTip, capacitate) — chei străine
- Zbor (codZbor, codAeropPlec, codAerop Sosire, oraPlecare, oraSosire, tarif, zisăpt)
- Rezervare (loc, codZbor, seriePas) — chei străine
- Pasager (nume, seriePas, nr. telefon)

Dependențe:

codAeroport → numeAeroport, țară
codTip → producător, model
codZbor → codAeropPlec, codAeropSos, oraPle, oraSos, tarif, zisăpt
seriePas → nume, nrtelefon
codZbor, loc → seriePas (un loc nu poate fi rezervat de 2 persoane)
codZbor, seriePas → loc (o persoană rezervă cel mult un loc la un zbor)

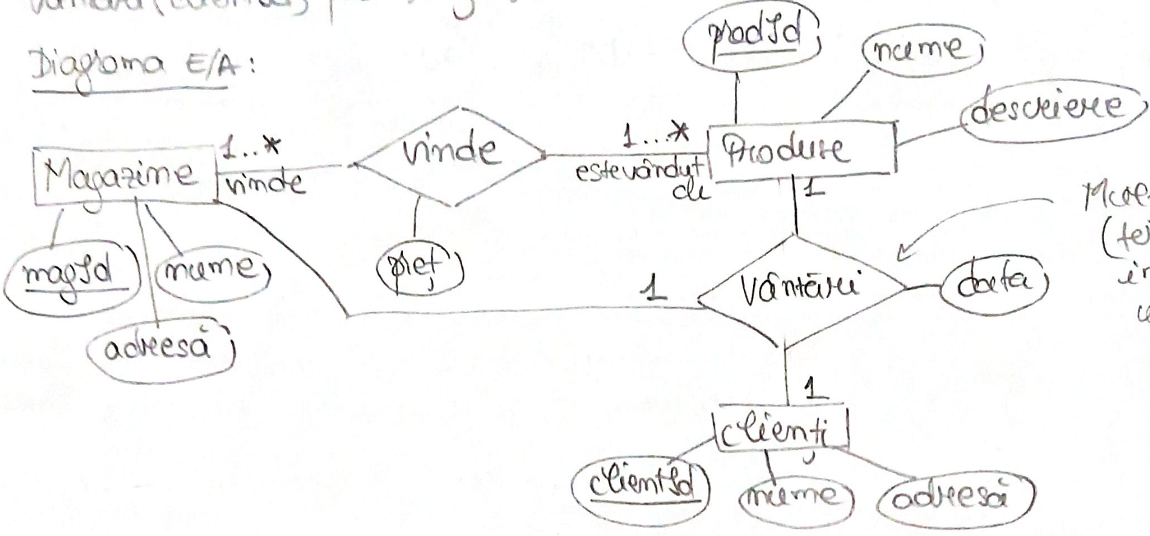
∃ o unică cheie: (codAeroport, codTip, codZbor, seriePas)

Pp. că schema satisf. 1NF.
∃ dependențe precum codAeroport → numeAeroport, țară, în care atributele 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:

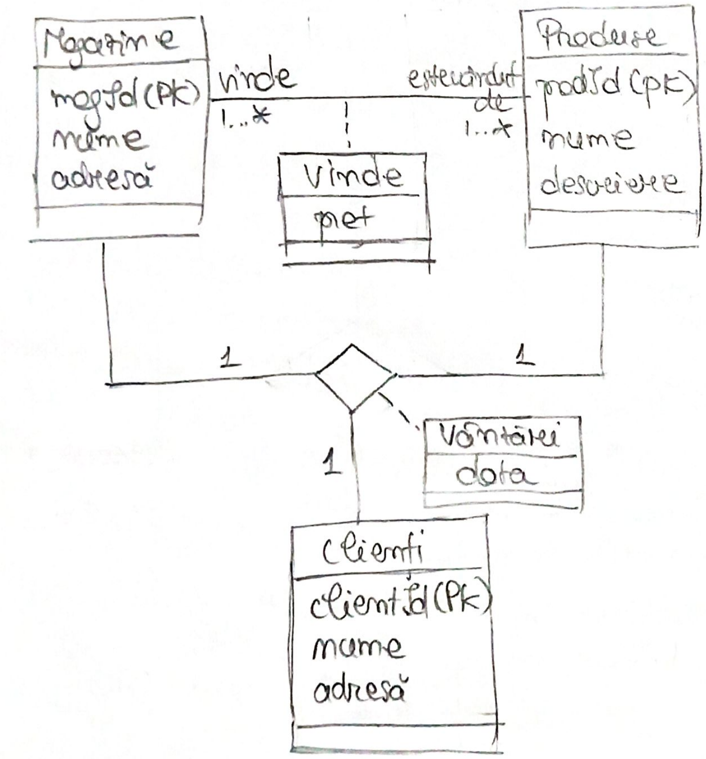
Magazine (magId, nume, adresă)
Produse (prodId, nume, descriere)
Vinde (magId, prodId, preț) ((magId, prodId) formează cheie primară => relația dintre Magazine și Produse este de tip mulți la mulți)
Clienți (clientId, nume, adresa)
Vânzări (clientId, prodId, magId, data)

Diagrama E/A:



Mulțimea-asociere (ternară) Vânzări implică faptul că un produs este vândut unui client într-un magazin la o anumită dată

Diagrama UML asociată:



(Obs: În general în UML nu se modelează asocieri cu grad > 2 și atribute proprii prin diagrame de clasă)