Modul Analiza

CUPRINS:

[1. Descrierea modelului ales si obiectivele aplicatie 2](#_Toc134651481)

[2. Diagramele bazei de date OLTP 2](#_Toc134651482)

[2.1 Diagrama Entiate-Relatie 3](#_Toc134651483)

[2.2 Diagrama Conceptuala 3](#_Toc134651484)

[3. Descrierea modului de distribuire 4](#_Toc134651485)

[4. Argumentarea deciziei de fragmentare a relatiilor 4](#_Toc134651486)

[4.1 Fragmentarea orizontala primara 5](#_Toc134651487)

[4.2 Fragmentarea orizontala derivata 6](#_Toc134651488)

[4.3 Fragmentarea verticala 6](#_Toc134651489)

[5. Verificarea corectitudinii fragmentarilor 6](#_Toc134651490)

[6. Argumentarea deciziilor de replicare 6](#_Toc134651491)

[7. Crearea schemelor conceptuale pentru bazele de date locale 6](#_Toc134651492)

[7.1 Bucuresti 7](#_Toc134651493)

[7.2 Provincie 7](#_Toc134651494)

[8. Constrangerile modelului 8](#_Toc134651495)

[8.1 Unicitate 8](#_Toc134651496)

# Descrierea modelului ales si obiectivele aplicatie

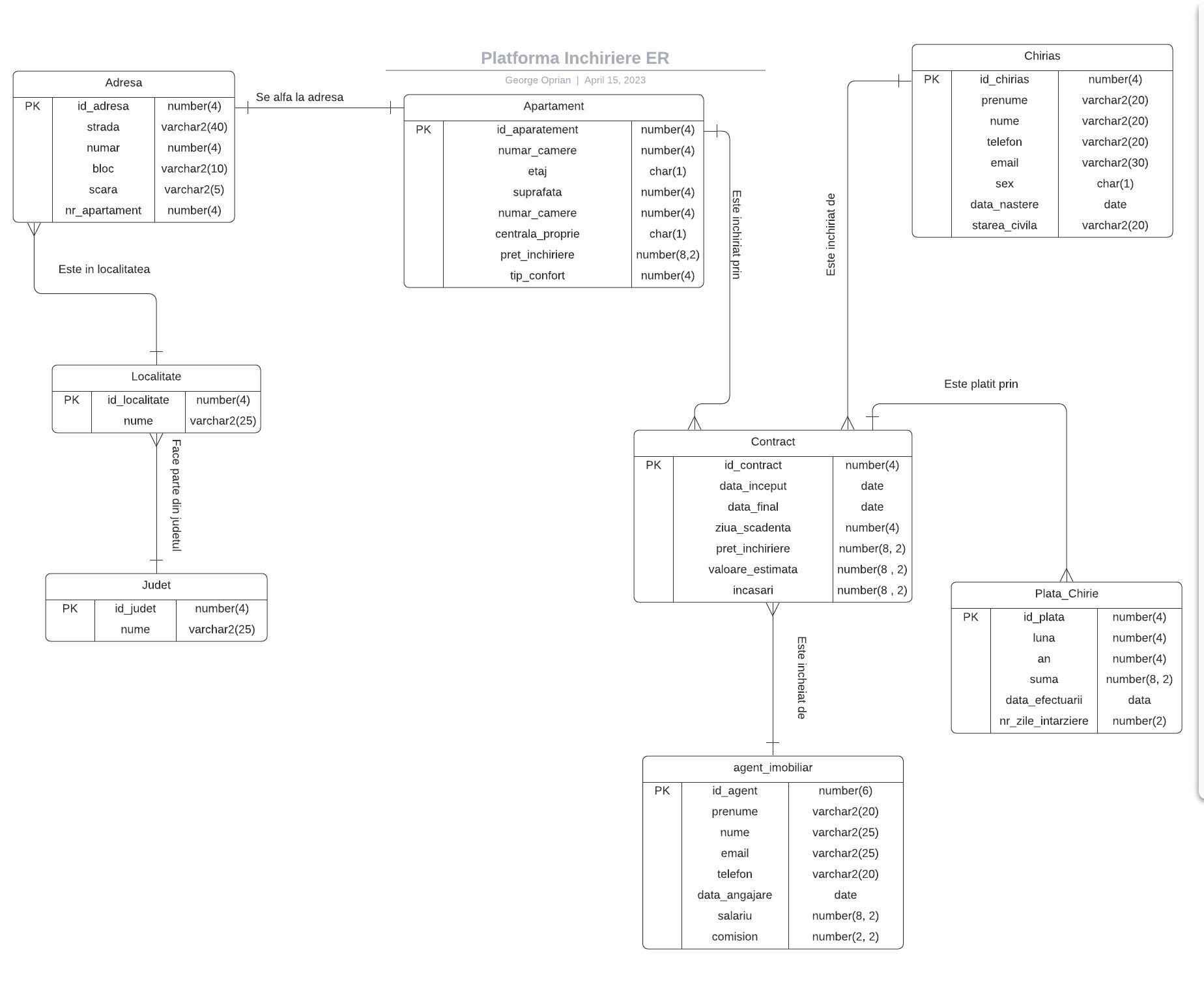
Pentru a indeplini cerintele proiectului am ales sa implementam o aplicatie de gestiune a unei agentii imobiliare care inchiriaza locuinte. In acest domeniu de activitate este nevoie de aplicatii de gestiune a proprietatilor doarece in ultimii ani cererea de inchiere de locuinte a crescut, din mai multe motive printe care putem enumera preturile destul de mari ale locuintelor, oamenii nu vor sa mai achizitioneze case doarece stilul lor de viata si locul de munca le permite sa schimbe locul in care locuiesc destul de des, sau poate doar doresc o locuinta temporara pana isi croiesc un rost in viata.

Astfel aplicatia noastra isi propune sa rezolve acesta problema prin gestionarea principalelor entitati prezente intr-o astfel de platforma: apartamente, chiriasi, angajati, contracte si plati.

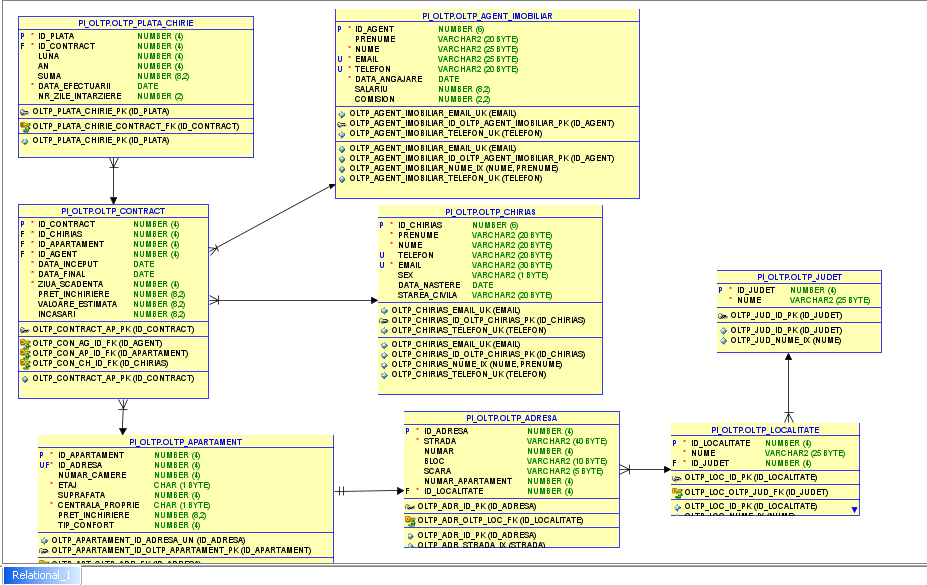
Deoarece agentia imobiliara gestionaza foarte multe apartamente din diverse orase ale tarii, cat si din capitala, si datorita faptului ca in capitala are un numar mare de apartamente inchiriate, a fost nevoie de o impartire a bazei de date centralizate a aplicatiei in doua baze de date locale: una pentru provincie si alta pentru Bucuresti.

# Diagramele bazei de date OLTP

## Diagrama Entiate-Relatie



## Diagrama Conceptuala



# Descrierea modului de distribuire

Am decis sa implementam o distribuire a bazei de date centralizate in doua baze de date locale in functie de localizarea apartamentului. Cele doua baze de date sunt definite pe aceeasi masina fizica. Astfel, am impartit baza de date centralizata in doua baze de date locale: Provincie si Bucuresti. Datele din cele doua baze de date urmand sa fie inglobate intr-o singura schema, nationala.

Cele 3 scheme sunt impartite pe doua baze de date fizice: modbd1 si modbd2. Urmand exemplul din laborator, pe modbd1 au fost definite schema pentru baza de date Bucuresti si National, fiecare avand utilizatori proprii, iar pe modbd2 a fost definita schema pentru baza de date Provincie, cu utilizatorul specific.

# Argumentarea deciziei de fragmentare a relatiilor

Pentru a putea defini cele doua baze de date locale, baza de date centralizata a trebuit supusa mai multor metode de fragmentare.

## Fragmentarea orizontala primara

Pentru a putea sparge tabelele curente in cel putin doua fragmente orizontale, am decis sa aplicam o fragmentare orizontala bazata pe localizarea apartamentelor. Astfel, am plecat de la idea ca apartamentele pot fi impartite in doua categorii: apartamente din Bucuresti si apartamente din provincie. Totusi, pentru a putea face aceasta impartire trebuie plecat de la judetul in care se afla apartamentul.

Astfel, fragmentarea orizontala trebuie sa inceapa cu fragmentare tabelei OLTP\_JUDET.

Aplicarea algoritmului COMP\_MIN:

**input**

R = OLTP\_JUDET

Pr = {nume=Bucuresti, nume<>Bucuresti} // Pr = {p1, p2}

**declare**

F: {} //multime de predicate simple completa si minimala

**output**

Pr' <- {nume=Bucuresti} // multime de predicate simple completă și minimala

Pr <- Pr \ {nume=Bucuresti}

F = {f1} // f1 = fragmenul lui p1

` **repeta**

//cautam un predicat pj (unde 1<j<=2) din Pr care partitioneaza fragmente fk din Pr' conform regulii 1

//Consideram j <- 2

k <- 2

Pr' <- Pr' <- {nume=Bucuresti, nume<>Bucuresti}

Pr <- Pr \ {p2} <- {}

F <- {f1, f2}

if exista p2 apartine lui Pr' nerelevant -> FALS

pana cand Pr' completa // dupa primul pas Pr' completa => algoritmul se termina

Deci multimea de predicate simpla si completa este **Pr'** = {nume=Bucuresti, nume<>Bucuresti}, **Pr** = {}, **F** = {f1, f2}, unde **f1 = multimea inregistrarilor cu nume=Bucuresti, f2 = multimea inregistrarilor cu nume<>Bucuresti**.

Prin urmare, vom avea doua fragmente orizontale ale relatiei OLTP\_JUDET: JUDET\_BUCURESTI si JUDET\_PROVINCIE.

## Fragmentarea orizontala derivata

Plecand de la fragmentare tabelului OLTP\_JUDET putem fragmenta si urmatoarele table care tin de localizarea unui apartament.

Astfel fragmentarea orizintala se propaga in urmatoarele tabele:

* OLTP\_LOCALITATE: LOCALITATE\_BUCURESTI si LOCALITATE\_PROVINCIE
* OLTP\_ADRESA: ADRESA\_BUCURESTI si ADRESA \_PROVINCIE
* OLTP\_APARTAMENT: APARTAMENT\_BUCURESTI si APARTAMENT\_PROVINCIE
* OLTP\_CONTRACT: CONTRACT\_BUCURESTI si CONTRACT\_PROVINCIE
* OLTP\_PLATA\_CHIRIE: PLATA\_CHIRIE\_BUCURESTI si PLATA\_CHIRIE\_PROVINCIE

## Fragmentarea verticala

Am ales sa fragmentam vertical tabelul OLTP\_AGENT\_IMOBILIAR deoarece face destul de usor o separare intre coloanele acestui tabel. Astfel am imparit coloanele in doua categorii: coloane ce tin de datele persoanale ale agentului si coloane care tin de salarizare.

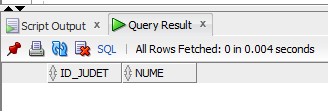
De adaugat @Petru

# Verificarea corectitudinii fragmentarilor

Pentru a verifica daca fragmentele au fost fragmentate corect trebuie verificate completitudinea, reconstructia si disjunctia pentru fiecare tabel.

OLTP\_JUDET:

* Completitudinea
  + select \*

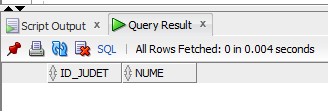
from modbd\_centralizat.oltp\_judet

minus

(select \* from JUDET\_BUCURESTI

union all

select \* from JUDET\_PROVINCIE@bd\_provincie);

* Reconstructia
  + JUDET\_BUCURESTI inclus in JUDET\_PROVINCIE
  + select \*

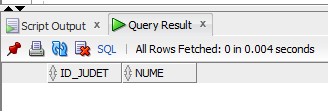
from modbd\_centralizat.oltp\_judet

minus

(select \* from JUDET\_BUCURESTI

union all

select \* from judet\_provincie@bd\_provincie);

* + JUDET\_PROVINCIE inclus in JUDET\_BUCURESTI

(select \* from JUDET\_BUCURESTI

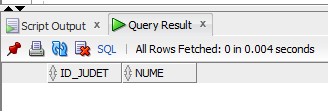
union all

select \* from JUDET\_PROVINCIE@bd\_provincie)

minus

select \*

from modbd\_centralizat.continents;

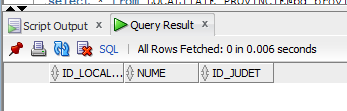
* Disjunctia
  + select \* from JUDET\_BUCURESTI

intersect

select \* from JUDET\_PROVINCIE@bd\_provincie

OLTP\_LOCALITATE:

* Completitudinea

select \*

from modbd\_centralizat.OLTP\_LOCALITATE

minus

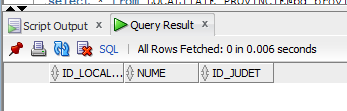
(select \* from LOCALITATE\_BUCURESTI

union all

select \* from LOCALITATE\_PROVINCIE@bd\_provincie);

* Reconstructia
  + LOCALITATE\_BUCURESTI inclus in LOCALITATE\_PROVINCIE

select \*

from modbd\_centralizat.OLTP\_LOCALITATE

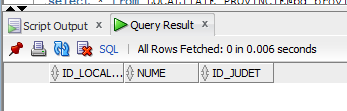
minus

(select \* from LOCALITATE\_BUCURESTI

union all

select \* from LOCALITATE\_PROVINCIE@bd\_provincie);

* + LOCALITATE\_PROVINCIE inclus in LOCALITATE\_BUCURESTI

(select \* from LOCALITATE\_BUCURESTI

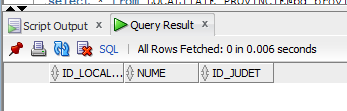
union all

select \* from LOCALITATE\_PROVINCIE@bd\_provincie)

minus

select \*

from modbd\_centralizat.OLTP\_LOCALITATE;

* Disjunctia

select \* from LOCALITATE\_BUCURESTI

intersect

select \* from LOCALITATE\_PROVINCIE@bd\_provincie;

# Argumentarea deciziilor de replicare

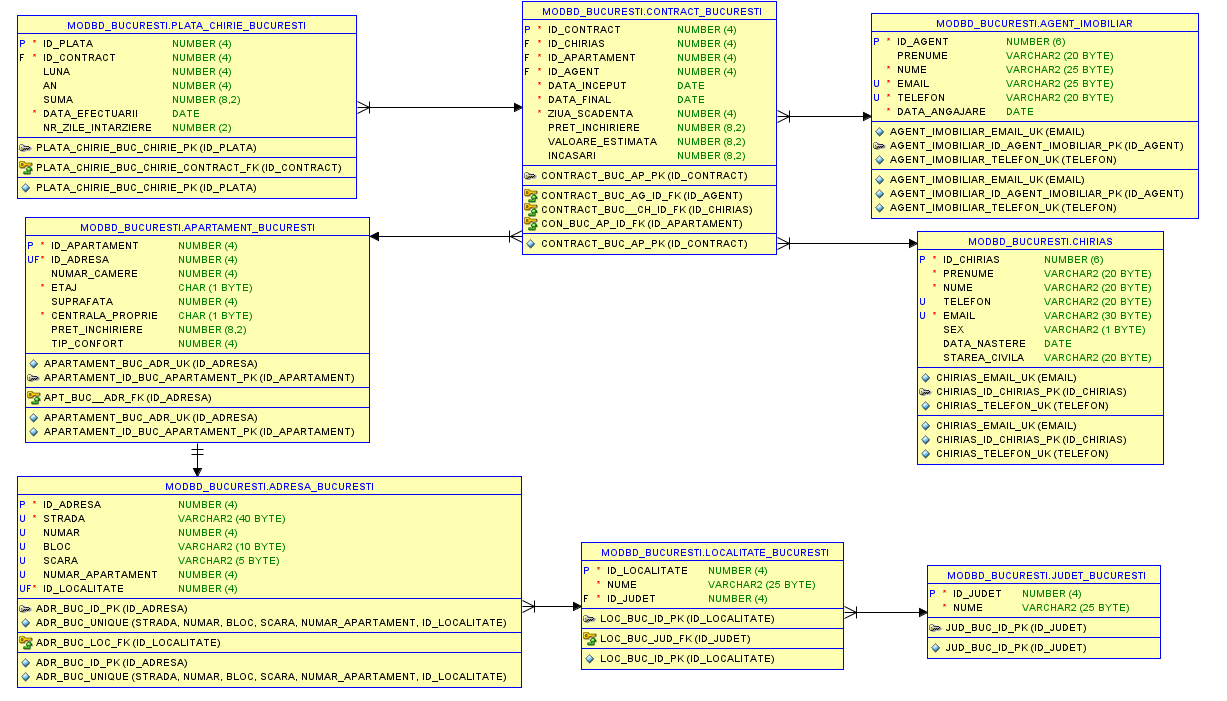
Am ales sa replicam datele din tabelele CHIRIAS si detaliile personale din AGENT\_IMOBILIAR, mai exact detaliile despre agent din bazele de date locale.

Principalul motiv pentru care am ales sa replica detaliile despre chirias a fost dat de faptul ca un chirias se poate muta dintr-un oras in altul, iar in cazul in care doreste sa inchirieze un alt apartament din alt oras este neplacut ca acesta sa isi introduca datele personale daca acestea nu s-au modificat.

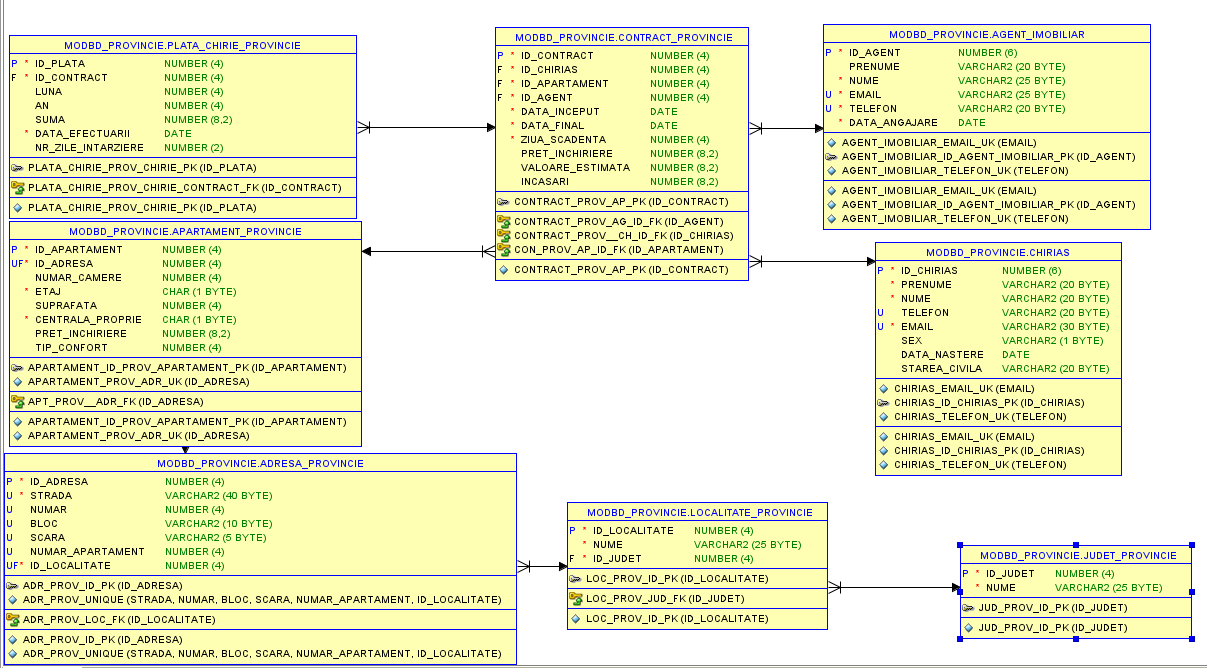
In cazul replicarii tabelului de agenti, am avut un motiv asemanator. Unii angajati pot fi relocati si nu este nevoie ca acestia sa isi introduca detaliile personale de fiecare data cand se relocheaza.

# Crearea schemelor conceptuale pentru bazele de date locale

## Bucuresti



## Provincie



# Constrangerile modelului

**Unicitate locala**

Pentru a pastra unicitatea datelor, constrangerile de unicitate au fost preluate din baza de date centralizata. Deoarece entitatile AGENT\_IMOBILIAR si CHIRIAS aveau constrangeri de unicitate pe artibutele EMAIL si TELEFON, a trebuit ca si aceste constrangeri sa fie migrate in bazele de date locale.

Pentru ca nu pot exista doua apartamente la aceeasi adresa a fost migrata constrangerea de unicitate pentru coloanele STRADA, NUMAR, BLOC, SCARA, NUMAR\_APARTAMENT, ID\_LOCALITATE ale tabelei centralizate OLTP\_ADRESA.

**Unicitatea globala pentru fragmente orizontale**

Pe tabelele ADRESA\_BUCURESTI si ADRESA\_PROVINCIE am definit, de asemenea constrangeri la nivel global gestionate prin triggeri pentru a asigura integritatea datelor.

**Unicitatea globala pentru fragmente verticale**

In cazul fragmentelor verticale, nu a fost nevoie sa definim constrangeri de unicitate sau integritate, pentru ca, in cazul tabelului AGENTI\_IMOBILIAR, nu avem nevoie de constrangeri intre fragmentele verticale.

**Cheie primara la nivel local**

Pentru toate tabelele de la nivel local, cheile primare au fost preluare de la entitatile din baza de date centralizata. Astfel s-au migrat toate aceste chei primare definite pe tabelele din centralizat.

**Cheie primara la nivel global**

Pentru a asigura uncitatea cheilor primare la nivel global, acestea au fost generate folosind secvente

Astfel, sa luam exemplul, tabelului APARTAMENT\_PROVINCIE. Pentru generarea id-urilor acestui tabel s-a folosit secventa BUCURESTI\_APARTAMENT\_SEQ care porneste de la numarul maxim de linii din baza de date centalizata ale tabelei OLTP\_APARTAMENT + 1, incrementandu-se din 2 in 2 pentru a lasa loc, la nivel global, si id-urilor din tabelul APARTAMENT\_BUCURESTI, care foloseste tot o secventa asemanatoarea pentru generarea id-urilor, paritatea id-urilor generate fiind diferita. Daca aplicam aceeasi metoda pentru toate relatiile fragmetnate orizontal, unicitate globala este garantanta in cazul in care generam id-uri folosind secvente.

**Cheile externe**

Cheile externe au fost, de asemenea, migrate si ele in bazele de date locale, in principal ele actionand doar pe tabelele din baza de date locala, nefiind nevoie ca doua entitati din baze de date diferite sa fie legate intre ele cu o cheie externa.

**Validarea constangerilor**

Validarea constrangerilor se face la inserarea/updatarea datelor in tabele. Astfel daca o constrangere not null, spre exemplu, este incalcata aceasta va declansa o eroare si va respinge operatia.

Dupa cum spuneam si mai sus, pentru a valida constangerile de unicitate la nivel global au fost definiti triggeri cu scopul de a verifica integritatea datelor.