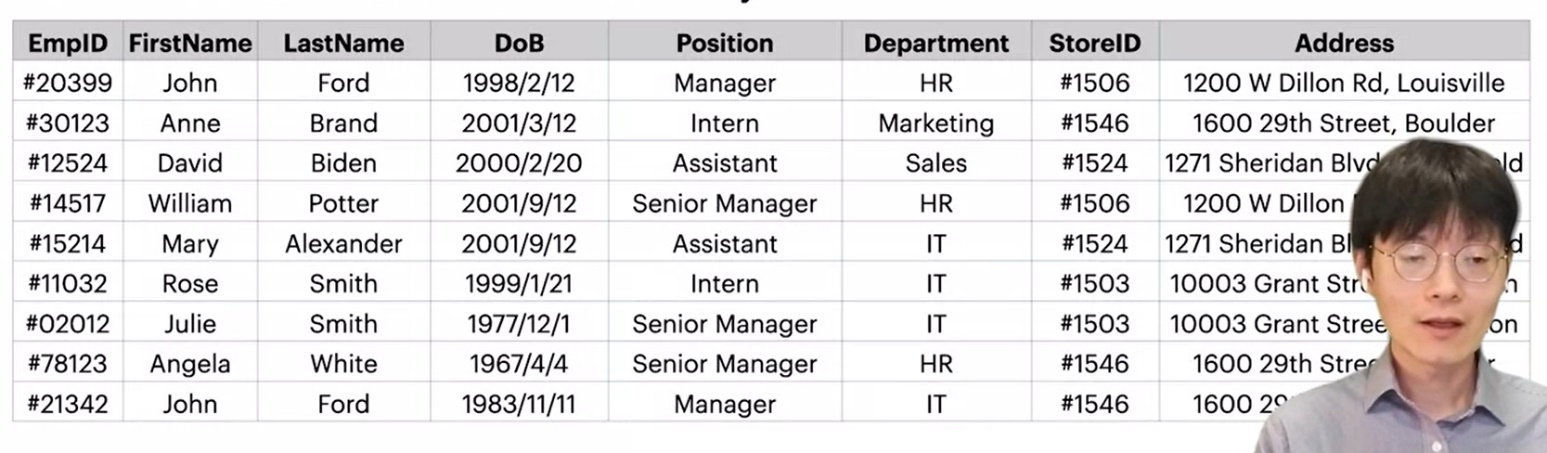
**Redutanta datelor**

* Redutanta datelor inseamna sa avem un set de date, in baza de date, stocate in multiple locuri si sa se repete inutil.
* Scopul de baza al unei baze de date relationara este de a grupa atributele in relatii in asa fel incat sa se minimalizeze redudanta datelor.
* Exemplu de redutanta a datelor:



In tabelul de mai sus, fiecare angajat lucreaza la un Store, si deci fiecare Store are un ID. Tabelul de sus e un tabel de Angajati. Exista deci si un tabel pentru fiecare Store, unde ele au ca atribut un StoreID. Totusi, in tabelul de sus pentru angajati, se cere sa punem ID la Store unde el lucreaza si Adresa la Store, insa Adresa la Store este deprisos, deoarece multi angajati lucreaza la aceleasi Store, si deci vor avea acelasi StoreID, si inca aceeasi adresa pentru Store la care lucreaza. Ca sa nu repetam StoreID nu avem ce face, dar adresa lor e redutanta, caci ea nu ofera nici-o informatie, asa cum putem gasi adresa la Store in tabelul de Stores. In plus, la inserarea acelorasi adrese de multe ori se pot face usor greseli. Mai rau, daca un Store isi modifica adresa, trebuie sa modificam adresele pentru toti angajatii din tabelul de sus ce lucreaza la acest Store, si de facut update ar fi mai greu,caci campurile cu address store la angajati tot ar trebui inlocuite, cand am putea pur si simplu modifica adresa in Tabelul Stores si atat.

Deci, pentru a rezolva reduntanta, doar excludem Address attribute din tabelul de sus.

* Probleme cu redutanta:
* Insert
* Update
* Delete

**Normalizarea**

* **Normalizare -** Tehnica de a crea relatii care evita posibila redutanta a datelor.
* **Normalizarea prevede ca sa utilizam numarul minim de attribute per tabel si sa nu repetam attribute existente intr-un tabel, in alt tabel, cu exceptia cazului cand ele sunt foreign key acolo.**
* Atributele care sunt strict legate intre ele se gasesc in aceeasi relatie.
* Se rezolva prin a muta anumite attribute sau a separa anumite relatii.
* Un prim pas al normalizarii, ar fi sa vedem ce date se tot repeta iar si iar in tabel, ar putea fi evitate cumva de tot repetat.
* Modelul relational creat initial ar putea sa nu fie intocmai bun. Noul model relationar obtinut dupa normalizare se cheama **Model Relational Normalizat.** Normalizarea se poate face chiar la crearea modelului relational.

**Dependenta Functionala**

* **Dependenta functionala descrie asocierile intre atributele *aceleasi* relatii.**
* NU PUTEM AVEA DEPENDENTA FUNCTIONALA INTRE RELATII
* Fie A si B doua multimi ce pot avea 1 sau mai multe atribute.

Daca A o determina pe B, atunci B este functional dependent de A.

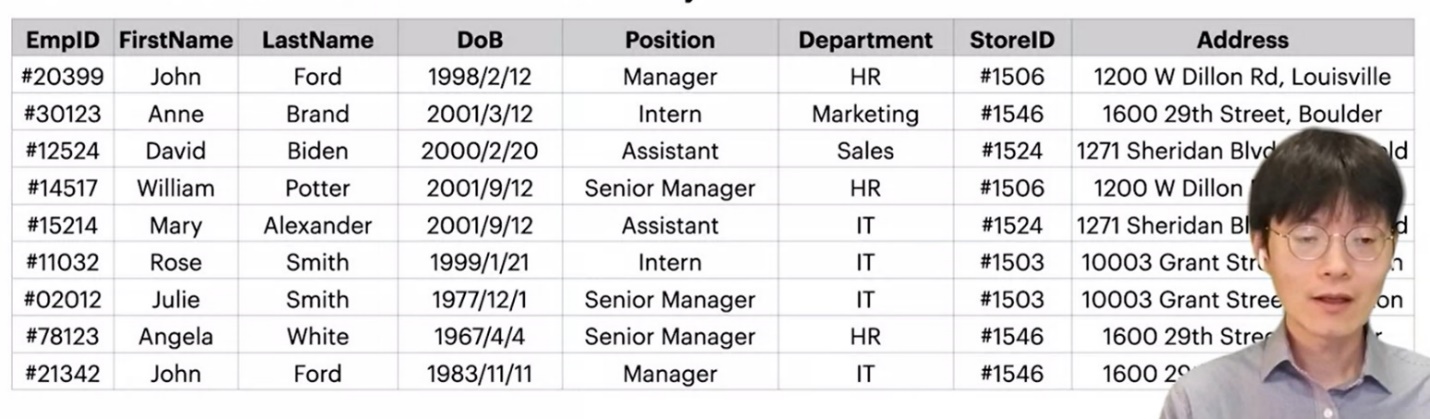
* **Daca A il determina pe B ca o asociere deterministica, atunci avem o dependenta functionala.**
* Dependenta functionala se noteaza **A -> B**

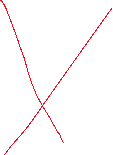
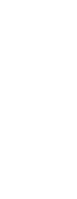
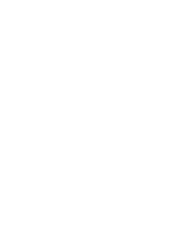
Asta s-ar traduce ca daca il stim pe A, il putem sti si pe B, sau daca il stim pe A, stim ca poate fi un unic B pentru acel A.

**In alte cuvinte, pentru o valoare a lui A poate exista o singura valoare a lui B, si putem sa il aflam anume si numai pe acel B cu A, fara a mai putea fi gasite si alte valori ale lui B pentru valaorea data a lui A. Nu conteaza ca aceasta valoare a lui A se mai gaseste in alte linii, e tot aceeasi, caci oricum va fi acelasi B peste tot. Asta inseamna ca putem gasi si mai multi de A prin tabel, dar numai daca acele obiecte au o singura valoare unica a lui B pentru acel A.**

**Nu conteaza daca si alte valori ale lui A pot avea acelasi B, ne intereseaza ca anume o anumita valoare a lui A sa aiba acelasi B;**

* Exemplu:





Daca stim EmpID, putem sti si care e FirstName, LastName si tot asa. Si ele sunt unice pentru acest EmpID, caci oricum fiecare obiect are un EmpID unic. Aici spunem ca EmpID determina FirstName si toate celelalte.

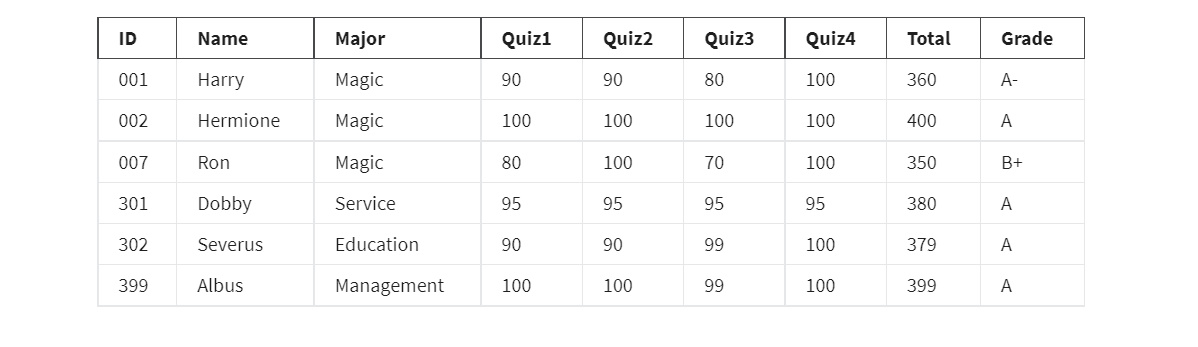
EmpID -> FirstName

EmpID -> LastName si tot asa

Dar nu putem spune ca FirstName determina ID, caci mai multi oameni pot avea acelasi FirstName, dar vor avea ID diferit.

Nici intre First Name + LastName -> Position nu exista, caci nu e garantat ca nu s-ar putea gasi 2 persoane cu acelasi Nume si Prenume, dar cu pozitii diferite.

De exemplu, Oras -> Raion ,e dependentaa functionala, caci daca avem de ex Singera la Oras, el poate aparea de oricate ori in tabel, oricum Raionul va fi Chisinau oriunde nu ar aparea Singera



Quiz1, Quiz2, Quiz3, Quiz4 -> Grade. Adica aceste 4 atribute pot sa ne gaseasca o valoare unica petntru Total.

**Tipuri de dependente functionala**

**Dependenta functionala ne ajuta la Normalizare**

* Full
* Partial
* Transitive

**Full Functional Dependency**

Fie A un set de atribute, adica A(a1,a2,a3,..) si B un atribut ce e dependent functional de A.

A(a1,a2,a3..) -> B

Daca nu exista nici-un subset(submultime) din A care determina de asemenea atributul B, atunci B este full dependent functional de A. Adica, A nu il poate determina pe B numai decat daca folosim absolut toate atributele multimii A! De aia si se cheama full, ca avem nevoie de toate atributele din A ca sa il determinam pe B.

Full functional dependenecy e minimalizarea lui Partial Functional Dependency, adica scoaterea atributelor din A de care nu avem nevoie sa il determinam pe B

**Partial Functional Dependency**

Fie A un set de atribute, adica A(a1,a2,a3,..) si B un atribut ce e dependent functional de A.

A(a1,a2,a3..) -> B

Daca exista un subset(submultime) din A care determina de asemenea atributul B, atunci B este partial dependent functional de A. Adica, daca exista in A un atribut sau mai multe, care pot separat, sau impreuna, sa il determine pe B, fara sa aiba nevoie si de alte atribute, atunci spunem ca B e partial functional dependent de A. Asta inseamna ca A are niste atribute inutile, ca si superkey si candidate key.

ID + NUME + PRENUME -> Data de nastere – e Partial, deoarece Nume si Prenume sunt in plus. Pentru o valoare a lui ID poate exista doar o singura valoare a datei, caci ID si asa e unic

**Transitive Functional Dependency**

Intr-o relatie, daca un atribut A il determina pe B si C si B il determina pe C, avem o dependenta functionala transitiva.

A ->B

B ->C

Si ca rezulta ca si A->C

B este ca un atribut de tranzit intre A si C

Sau A,B -> D,E,F,G

E->G

Deci A,B -> D,E,F,G tot e transitiva caci A,B il determina pe E si E pe G si nu conteaza ca si A,B il pot determina tot pe G singure.

**Key vs Functional Dependency**

* Canidate key si Primary Key sunt atribute unice pentru fiecare obiect, care garanteaza ca utilizandu-le, vom gasi un singur unic obiect. Adica, nu ar putea exista un alt obiect ce sa aiba aceeasi valoare a atributului keii pe care o mai au si altele.
* Functional Dependency nu inseamna ca doar un singur obiect are valorile lui A. A il determina pe B, adica folosind o anumita valoare a lui A, putem gasi o singura valoare a lui B. Nu conteaza ca o anumita valoare din A o au mai multe obiecte, important e ca anume aceeasi valore a lui A de la alte obiecte sa o determine tot pe aceeasi valoare a lui B.
* Intre primary key/candidate key si B nu mereu exista dependenta functionala.
* Daca A -> B full functional dependency, nu inseamna neaparat ca A e primary sau canditate key, asa cum o anumita valoare a lui A poate aparea de mai multe ori in acelasi tabel

**Procesul de Normalizare**

* **Normalizare -** Tehnica de a crea relatii care evita posibila redutanta a datelor.
* **Normalizare –** Proces de a converti o relatie de la o forma mai putin restrictiva la una la o forma mai restrictiva.
* Nivelul de restrictie e numit **Normal Form**

**Tipuri de Normal Form**

* **Unnormalized form(UNF):**

Nu e prezenta nici-o restrictie in tabel. Sunt prezente date ce se repeta in tabel inutil si o celula are chiar si 2 si mai multe valori, care iar duc la repetari de date.

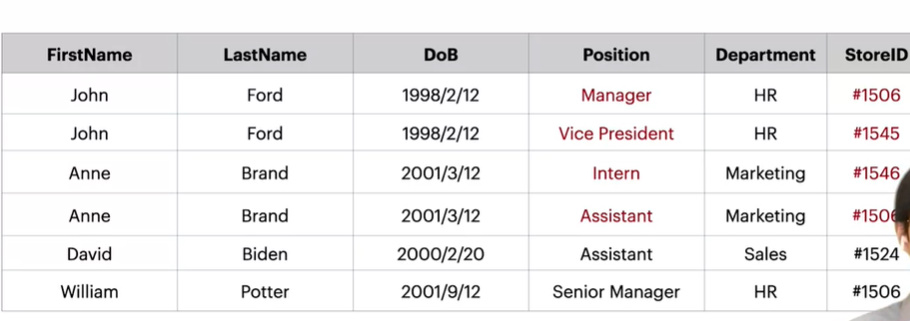


Aici in tabel sunt 2 coloane in care se gasesc cate 2 valori in celula.

* **First Normal Form(1NF)**

Tabelul nu poate contine celule in care se gasesc 2 sau mai multe valori! Doar o valoare per celula e permisa.

Pentru a rezolva problema de la Unnormalized Form de sus, si ca sa o trecem la 1NF, cream cate o linie pentru fiecare valoare din cele ce sunt mai multe intr-o celula.



Sau am putea pune limita ca fiecare angajat sa aiba doar 2 pozitii, si punem ca atribute Position1, StoreID1, Position2, StoreID2

* **Second Normal Form(2NF)**

Conditii:

1. Trebuie deja sa fie in 1NF(sa nu permita mai multe valori intr-o celula)
2. Fiecare non-primary key atribut este full functional dependent de primary key. Daca e partial functional dependent, nu mai e 2NF.

Non-primary key atribut este orice atribut ce nu e primary key.Aici avem situatia cand A -> B, unde A e primary key si B nu e primary key. Cu alte cuvinte, pentru o valoare a atributului A care e primary key, exista o singura valaore a lui B. Cand A e simple primary key, acest lucru e evident, dar daca A e formata din mai multe atribute, inseamna ca toate acele atribute **doar impreuna** pot determina o singura valoare a unui alt atribut non primary key.

Ex: R (A, B, C, D, E, F, G) A, B, C – primary key

A,B,C -> D,E,F,G

A -> D

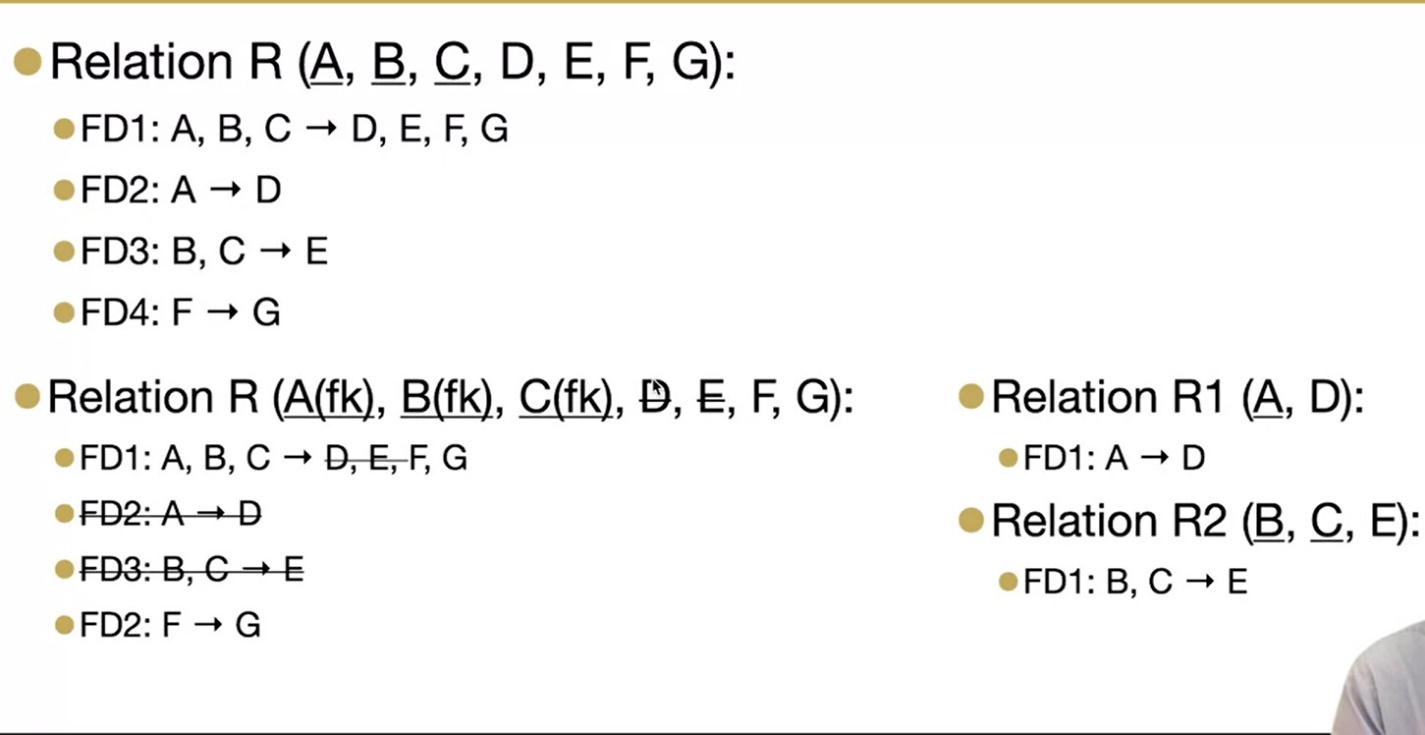
B,C -> E

F->G

Nu e 2NF! D si E nu sunt full dependente de primary key.

Pentru a trece de la 1NF la 2NF trebuie sa eliminam dependenta functionala partiala. Acest lucru il facem asa:

1. Setam fiecare atribut din cele ce sunt primary key si care participa in dependenta functional partiala ca foreign key si eliminam atributele care creaza o dependenta functionala partiala(D,E)
2. cream Tabele noi pentru fiecare dependenta functionala partiala eliminata, unde primary key vor fi atributele din fosta primary key din tabelul principal de care e dependent atributul.



Fie ca avem R(Nume, Prenume, PhoneNumber, Country, Address)

Nume, Prenume, PhoneNumber -> Country, Address

PhoneNumber -> Country

A doua e o dependentaa functionala partiala, deoarece putem sti tara si fara sa avem nevoie de Nume si Prenume. In asa caz, excludem cumva existenta dependentei functional partiale, prin eliminarea atributului Country, prin setarea lui PhoneNumber in tabelul original ca foreign key si cream un nou tabel unde PhoneNumber e primmary key si un alt atribut e country:

R(Nume, Prenume, PhoneNumber(FK), Address)

G(PhoneNumber, Country)

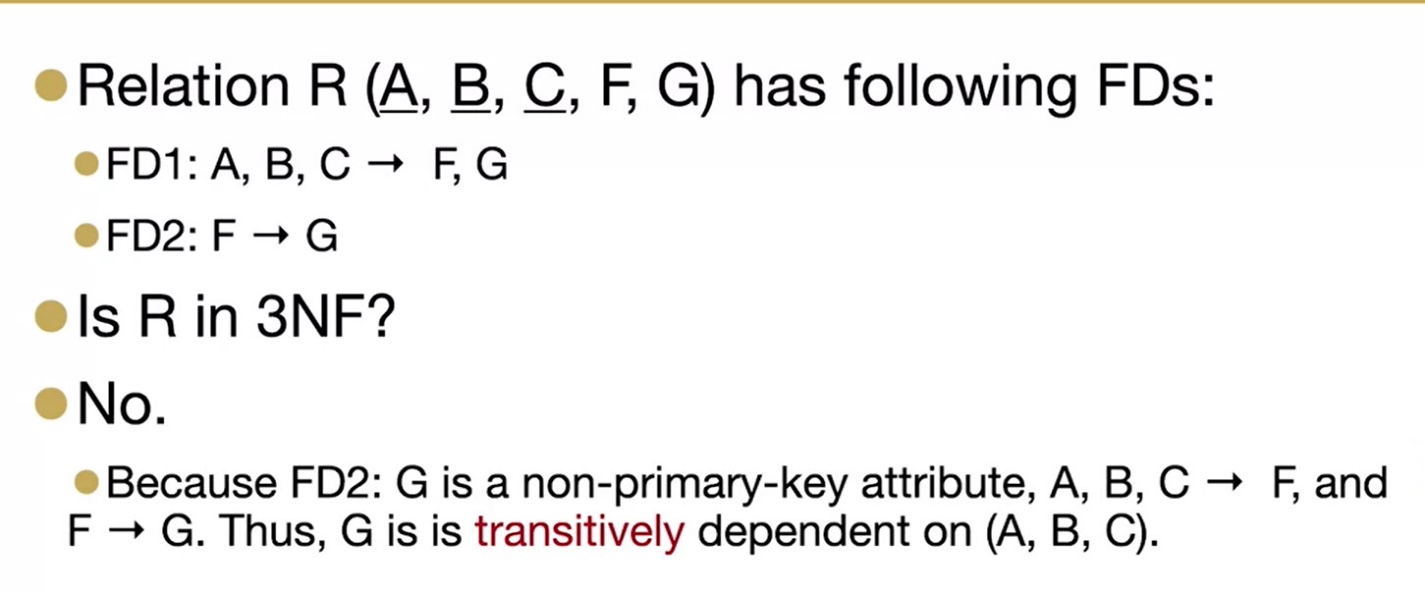
Asa dar acum avem doar:

Nume, Prenume, PhoneNumber -> Adress

* **Third Normal Form(3NF)(Minima necesara unei baze de date)**

Conditii:

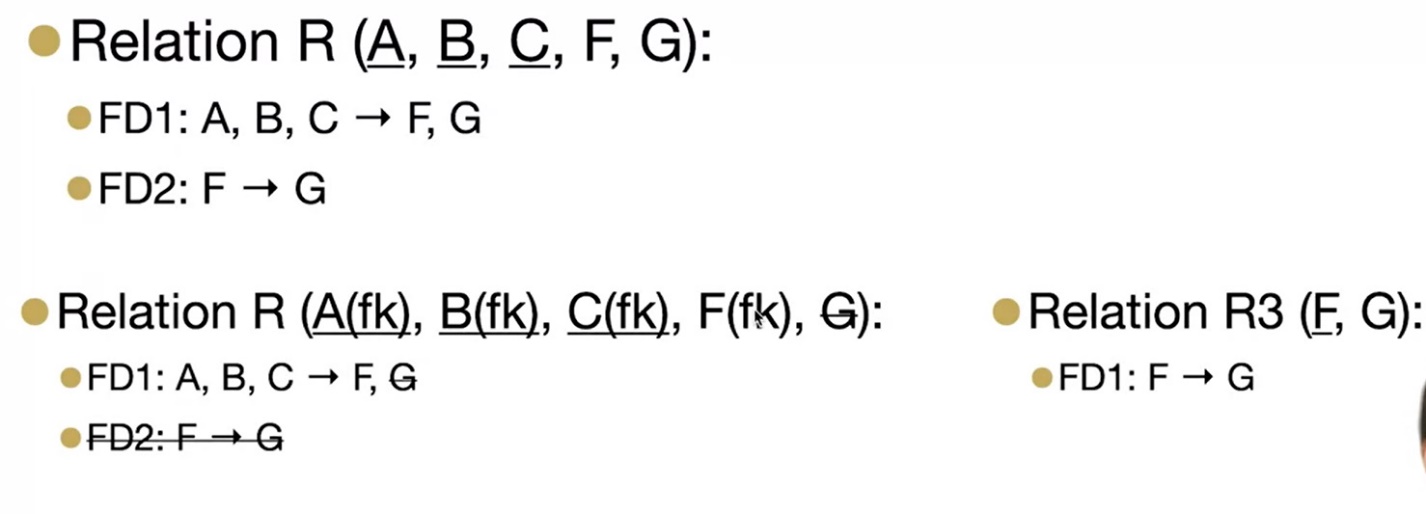
1. Este in 2NF
2. Fiecare non primary key atribut nu este transitiv functional dependent de primary key.



Deci, nu trebuie sa fie un alt atribut, decat cele din primary key, care sa poata determina un alt atribut din relatie.

Pentru a trece de la 2NF la 3NF, procedam tot cam ca trecerea de la 1NF la 2NF:

1. Eliminam asocierea care creaza dependenta functionala tranzitiva prin eliminarea din tabelul original a atributului care e dependent si setam atributul de care acesta e dependent ca foreign key
2. Cream un nou tabel in care acest Foreign key e primary key si atributul dependent de el e un atribut simplu.



**Si la 2NF si la 3NF noile tabele tot trebuie verificate!**

* **Higher Normal Form BCNF, 4NF, 5NF**