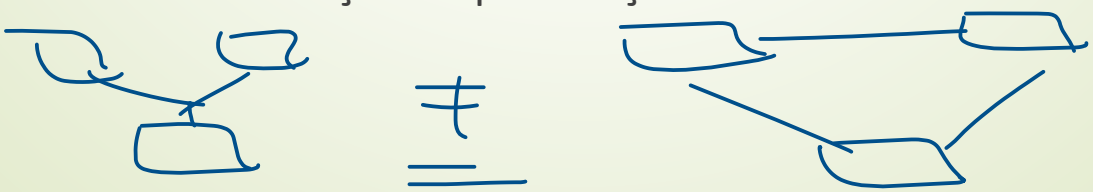
BAZE DE DATE

CURS 10

Forma normală 5 (FN5)

- FN5 îşi propune eliminarea redundanţelor care apar în relaţii *m:n* dependente.
  - În general, aceste relaţii nu pot fi descompuse.
  - S-a arătat că o relaţie de tip 3 este diferită de trei relaţii de tip 2. Există totuşi o excepţie, şi anume, dacă relaţia este ciclică
- **Intuitiv**, o relaţie *R* este în forma normală 5 dacă şi numai dacă:
  - relaţia este în FN4;
  - nu conţine dependenţe ciclice.



Forma normală 5 (FN5)

- Dependenţa funcţională şi multidependenţa permit descompunerea prin proiecţie, fără pierdere de informaţie, a unei relaţii în **două relaţii**.
- Regulile de descompunere (FN1 – FN4) nu dau toate descompunerile posibile prin proiecţie ale unei relaţii.
- Există relaţii care nu pot fi descompuse în două relaţii dar pot fi descompuse în **trei, patru sau mai multe relaţii fără a pierde informaţii**.
- Pentru a obţine descompuneri L-*join* în trei sau mai multe relaţii, s-a introdus conceptul de ***join*-dependenţă sau dependenţă la compunere** (JD).

Forma normală 5 (FN5)

- Fie $\{R_1, R_2, ..., R_p\}$ o mulţime de scheme relaţionale care nu sunt disjuncte şi a căror reuniune este R.
- R satisface ***join*-dependenţa** $*\{R_1, R_2, ..., R_p\}$ dacă la fiecare moment are loc egalitatea:

  $$R = \text{JOIN}(\Pi_{\alpha 1}(R), \Pi_{\alpha 2}(R), ..., \Pi_{\alpha p}(R))$$

  unde α_k reprezintă mulţimea atributelor corespunzătoare lui $R_k (1 \le k \le p)$.
- *Join*-dependenţa $*\{R_1, R_2, ..., R_p\}$ are loc în R, dacă $R_1, R_2, ..., R_p$ este o descompunere L-*join* a lui R.
- Pentru $p = 2$ se regăseşte multidependenţa.
- O *join*-dependenţă $*\{R_1, R_2, ..., R_p\}$ în care una dintre R_i este chiar R, defineşte o *join*-dependenţă trivială.

Forma normală 5 (FN5)

- *Join*-dependenţa **generalizează multidependenţa**.
- Într-adevăr, multidependenţa $X \rightarrow\rightarrow Y$ în relaţia $R(X, Y, Z)$ (deci şi $X \rightarrow\rightarrow Z$), corespunde *join*-dependenţei $*\{X \cup Y, X \cup Z\}$.
- Invers, *join*-dependenţa $*\{R1, R2\}$ corespunde multidependenţei $R1 \cap R2 \rightarrow\rightarrow R1 - (R1 \cap R2)$.
- Formal, o relaţie R este în FN5 dacă şi numai dacă orice *join* dependenţă $*\{R_1, R_2, ..., R_p\}$ care are loc în R fie este trivială, fie conţine o supercheie a lui R (adică, o anumită componentă R_i este o supercheie a lui R).
- Cu alte cuvinte, o relaţie R este în FN5 dacă orice *join*-dependenţă definită pe R este implicată de cheile candidat ale lui R.

Forma normală 5 (FN5)

- Între mulţimile de atribute *X*, *Y* şi *Z* din cadrul relaţiei *R* există o *join* dependenţă dacă există multidependenţe între fiecare dintre perechile de mulţimi (*X*, *Y*), (*Y*, *Z*) şi (*X*, *Z*).
- Aducerea în FN5 prin eliminarea join dependenţelor!

Forma normală 5 (FN5)

- Între mulţimile de atribute *X*, *Y* şi *Z* din cadrul relaţiei *R* există o *join* dependenţă dacă există multidependenţe între fiecare dintre perechile de mulţimi (*X*, *Y*), (*Y*, *Z*) şi (*X*, *Z*).
- Aducerea în FN5 prin eliminarea join dependenţelor!
- Exemple – suport curs. → vezi adnotării de la finalul documentului

Concluzii NORMALIZARE

1. FN1 → FN2 elimină redundanţele datorate dependenţei netotale a atributelor care nu participă la o cheie, faţă de cheile lui *R*. Se suprimă dependenţele funcţionale care nu sunt totale.
2. FN2 → FN3 elimină redundanţele datorate dependenţei tranzitive. Se suprimă dependenţele funcţionale tranzitive.
3. FN3 → BCNF elimină redundanţele datorate dependenţei funcţionale. Se suprimă dependenţele în care partea stângă nu este o supercheie.
4. BCNF → FN4 elimină redundanţele datorate multidependenţei. Se suprimă toate multidependenţele (non-cheie) care nu sunt şi dependenţe funcţionale.
5. FN4 → FN5 elimină redundanţele datorate dependentei ciclice. Se suprimă toate *join*-dependenţele care nu sunt implicate de o cheie.
6. BCNF, FN4 şi FN5 corespund la regula că orice determinant este o cheie, dar de fiecare dată dependenţa cu care se defineşte determinantul este alta şi anume dependenţa funcţională, multidependenţa sau *join*-dependenţa).
7. Descompunerea unei relaţii FN2 în FN3 conservă datele şi dependenţele, pe când descompunerea unei relaţii FN3 în BCNF şi, respectiv, a unei relaţii BCNF în FN4 conservă doar datele.

DENORMALIZARE

- Procedura de normalizare elimină redundanțele prin efectuarea unor proiecții, DAR NU toate redundanțele pot fi eliminate în acest mod.

=>Uneori este necesară **denormalizarea** care presupune:

- Mărirea redundanței;
- Reducerea numărului de join-uri care trebuie efectuate => micșorare timp de execuție!

DENORMALIZARE

- Ideile normalizării sunt utile în proiectarea BD, dar nu sunt obligatorii!
- Dependența și normalizarea sunt de natură semantică (cu alte cuvinte, se referă la ceea ce înseamnă datele).
- In schimb, algebra relațională și calculul relațional (limbajele SQL) se referă doar la valorile efective ale datelor și în multe cazuri nu necesită mai mult decât FN1.

FN3

DENORMALIZARE

A) Obiectivul denormalizării constă în **reducerea numărului de join-uri** care trebuie efectuate pentru rezolvarea unei interogări, prin realizarea unora dintre acestea în avans, ca făcând parte din proiectarea bazei de date.

DENORMALIZARE

B) Conceptul de denormalizare suferă de un număr de probleme binecunoscute.

- Odată începută denormalizarea, nu este clar unde trebuie să se oprească.
- Nu se mai lucrează cu relații normalizate și astfel pot apărea anomaliile pe care normalizarea le corectează.
- Un design fizic poate fi bun pentru anumite aplicații, dar prost pentru altele (denormalizarea la nivelul fișierelor stocate).

DENORMALIZARE

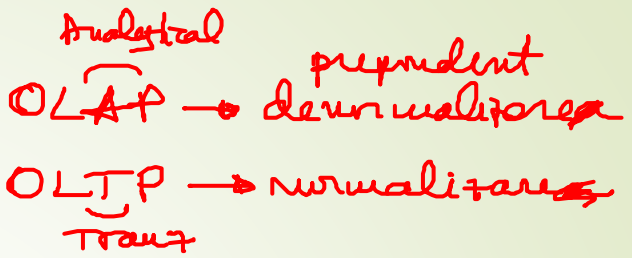
Exemplu:

- Se poate presupune că fiecare tabel corespunde unui anumit fișier stocat și că fiecare fișier stocat este format dintr-o mulțime contiguă fizic de înregistrări stocate, câte una pentru fiecare tuplu din tabel.
- Se presupune că informațiile (join-ul) despre creatori, vestimentații și accesorii se reprezintă printr-un tabel și, prin urmare, un fișier stocat. In această structură fizică interogarea « Obținerea informațiilor despre creatorii care oferă vestimentații cu accesorii din margele » se rezolvă cu ușurință.

DENORMALIZARE

- Interogarea « Obținerea informațiilor despre creatorii din Sibiu » va prezenta performanțe mai reduse în această structură fizică, decât dacă am fi menținut trei tabele de bază pentru cele 3 entități și, prin urmare, 3 fișiere stocate fizic separat.
- In acest ultim design, toate înregistrările despre creatori vor fi fizic contigue, în timp ce în primul design ele sunt dispersate fizic într-o zonă largă și, prin urmare, vor necesita mai multe operații I/O.

DENORMALIZARE



Când este utilă denormalizarea?

- Ca o regulă empirică, se poate afirma că, dacă performanțele nu sunt satisfăcătoare și relația are o rată de reactualizare scăzută, dar o rata a interogarilor foarte ridicata, denormalizarea poate constitui o opțiune viabilă.
- Nu există reguli fixe pentru stabilirea situatiilor in care este indicată denormalizarea relațiilor.
- În general, denormalizarea înseamnă considerarea dublării atributelor și grupării relațiilor
  - Dublarea (duplicarea) atributelor sau gruparea relațiilor are ca scop reducerea numărului de join-uri necesare pentru efectuarea unei interogări

Exemplu de denormalizare

- FILIALA (cod_filiala#, strada, zona, oras, cod_postal, telefon)
- Relația nu este în FN3 deoarece cod_postal → {zona, oras}
- Aplicăm FN3 și se obține:
  - FILIALA1(cod_filiala#, strada, cod-postal, telefon)
  - COD_P(cod_postal#, zona, oras)
- Nu este convenabil, deoarece rareori vom accesa adresa filialei, fără informații la zonă și oraș. Prin urmare, în acest caz putem prefera varianta denormalizată (cea aflată în FN2)

Cazuri de denormalizare

- Considerarea datelor derivate
- Combinarea relațiilor de tip 1:1
- Duplicarea atributelor care nu sunt chei în relații 1:M
- Tabele de căutare
- Duplicarea atributelor cheii externe într-o relație 1:M pentru simplificarea join-urilor
- Duplicarea atributelor în relațiile de tip M:N, pentru reducerea join-urilor

Cazuri de denormalizare

Duplicarea atributelor care nu sunt chei în relații 1:M

```
SELECT p.*, pp.nume
FROM   proprietate_de_inchiriat p, proprietar pp
WHERE  p.cod_proprietar = pp.cod AND cod_filiala = 'S3';
```

- Dacă se va duplica atributul nume în relația *proprietate_de_inchiriat*, interogarea devine:

```
SELECT p.*
FROM   proprietate_de_inchiriat p
WHERE  cod_filiala = 'S3';
```

- Avantaje sau dezavantaje? Depinde de problemele care pot apărea.

Cazuri de denormalizare

Tabele de căutare (de referință)

- Acestea conțin, de obicei, un cod și o descriere (și / sau denumire)
- Le întâlnim în forma normalizată a bazei de date
- De exemplu, se poate defini un tabel de căutare pentru tipul de proprietate și modifica tabelul *proprietate_de inchiriat* astfel:

TIP_PROPRIETATE(cod_tip#, descriere)

PROPRIETATE_DE_INCHIRIAT(cod#, strada, zona, oras, cod_postal, cod_tip, nr_camere, chirie, cod_proprietar, cod_filiala, nr_personal)

Cazuri de denormalizare

Tabele de căutare (de referință) - continuare

- Avantaje:
  - Dacă este modificată descrierea, atunci se va modifica o singură dată, în tabelul de căutare
  - Se reduce dimensiunea relației de la capătul M al relației (PROPRIETATE_DE_INCHIRIAT)
- Varianta de pe slide-ul anterior este în FN3
- Dacă se consideră că accesarea descrierii tipului de proprietate are loc frecvent odată cu accesarea informațiilor despre proprietate, atunci putem denormaliza astfel:

~~TIP_PROPRIETATE(cod_tip#, descriere)~~

PROPRIETATE_DE_INCHIRIAT(cod#, strada, zona, oras, cod_postal, descriere_tip, nr_camere, chirie, cod_proprietar, cod_filiala, nr_personal)

Cazuri de denormalizare

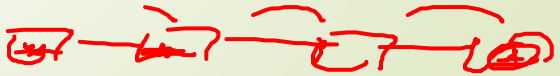
Duplicarea atributelor cheii externe într-o relație de tip 1:M

- Obiectiv: simplificarea join-urilor
- Relațiile pot să nu fie învecinate în diagramă
- **Exemplu**: Să se enumere proprietarii de proprietăți de închiriat dintr-o filială.

```
SELECT pp.nume
FROM   proprietate_de_inchiriat p, proprietar pp
WHERE  p.cod_proprietar = pp.cod AND cod_filiala = 'S3';
```

- Dacă se duplică cheia externă *cod_filiala* în relația PROPRIETAR, adică se introduce o relație directă între FILIALA și PERSONAL, atunci cererea devine:

```
SELECT pp.nume
FROM   proprietar pp
WHERE  cod_filiala = 'S3';
```



Cazuri de denormalizare

Duplicarea atributelor cheii externe într-o relație de tip 1:M (continuare)

- **Atenție!** Sunt necesare constrângeri suplimentare asupra cheilor externe. De exemplu, dacă un proprietar ar închiria prin mai multe filiale atunci modificările nu mai sunt valabile.
  - Prezența unui atribut multiplu ar însemna non-FN1.
- Observație: Singurul motiv pentru care relația PROPRIETATE_DE_INCHIRIAT conține atributul *cod_filiala* constă în faptul că este posibil ca o proprietate să nu aibă alocat un membru de personal, mai ales la început, atunci când este preluată de către agenție.

Cazuri de denormalizare

Duplicarea atributelor în relațiile de tip M:N

- Presupunem că relația M:N dintre CHIRIAS și PROPRIETATE_DE_INCHIRIAT a fost descompusă prin introducerea relației intermediare VIZITARE.
- **Exemplu:** Care sunt chiriașii care au vizitat proprietăți, dar mai au de făcut comentarii asupra unora dintre ele? Personalul de la agenție are nevoie de atributul *strada* atunci când discută cu chiriașii.

```
SELECT p.strada, c.*, v.data
FROM   chirias c, vizitare v, proprietate_de_inchiriat p
WHERE  v.cod_proprietate = p.cod AND c.cod =
v.cod_chirias AND comentarii IS NULL;
```

Cazuri de denormalizare

Duplicarea atributelor în relațiile de tip M:N (continuare)

- Dacă se introduce atributul strada în relația VIZITARE, atunci cererea devine:

```
SELECT v.strada, c.*, v.data
FROM   chirias c, vizitare v
WHERE  c.cod = v.cod_chirias AND comentarii IS NULL;
```

$X, Y, Z \quad X \twoheadrightarrow Y \longrightarrow X \twoheadrightarrow Z$

FN4

Ex. 1: CURSURI(Curs#, Prof#, Carte#)

Regula modelului: Un curs este predat folosind anumite carti.

Curs	Prof	Carte
1	P1	B1
1	P1	B2
1	P2	B1
1	P2	B2

1 ↠ P1, 1 ↠ P2, P1 ↠ B1, P2 ↠ B2

1	P2	B1
1	P1	B2

Curs → Carte ⟺ Curs → Profesor

Curs nu este CP ⟹ CURSURI nu este în FN4.

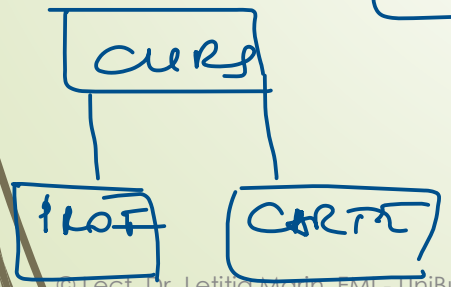
Aducere la FN4: CURSURI1 (Curs#, Carte#)
CURSURI2 (Curs#, Profesor#)

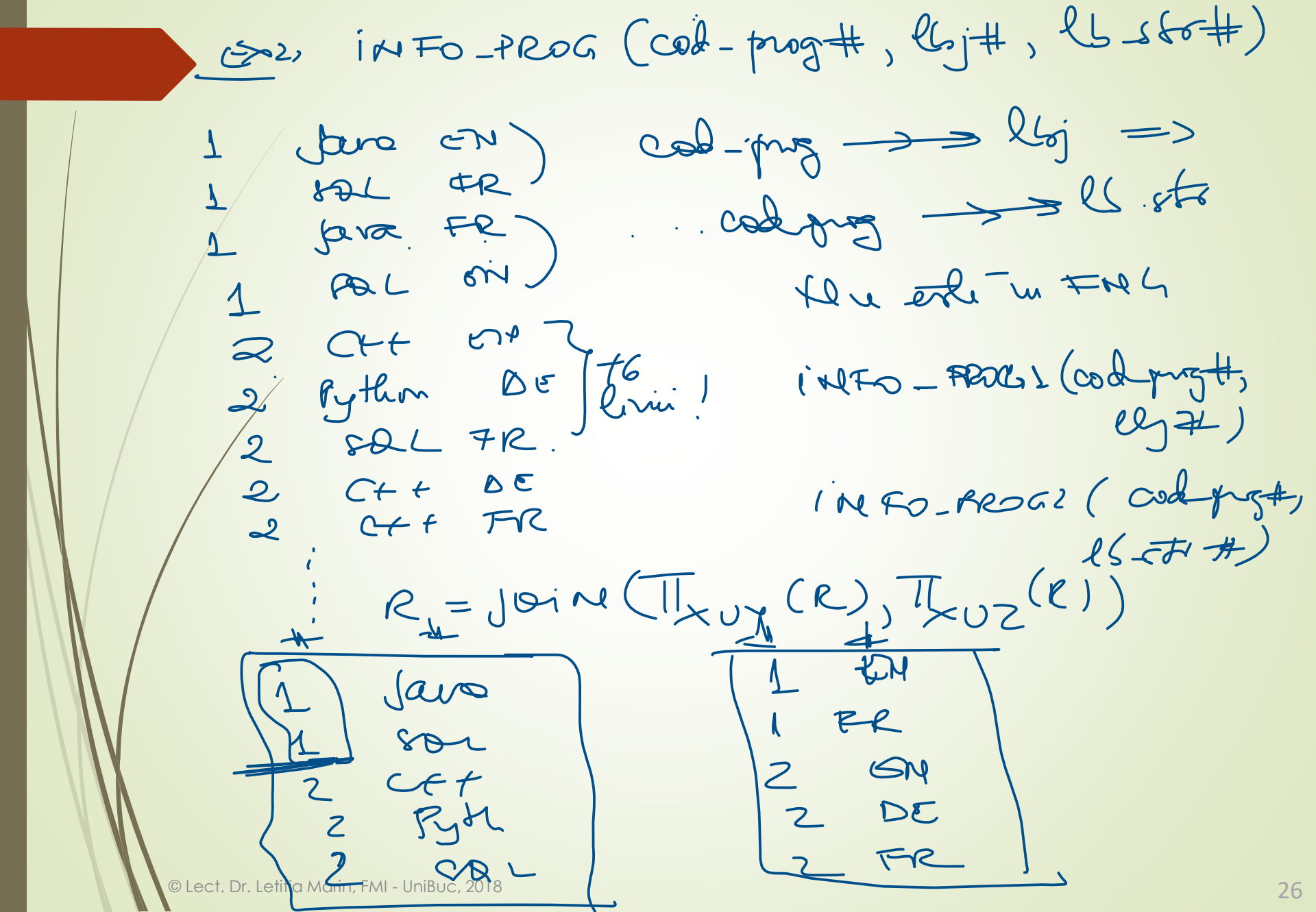
C1

1	B1
1	B2

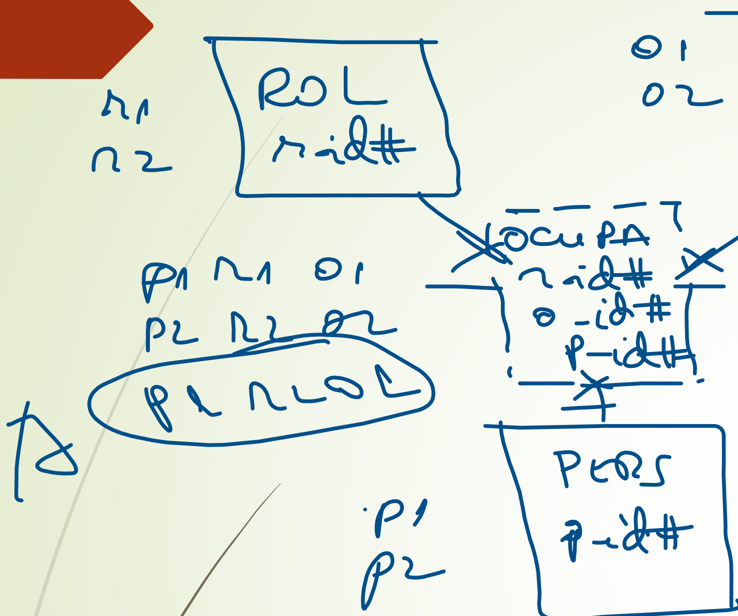
C2

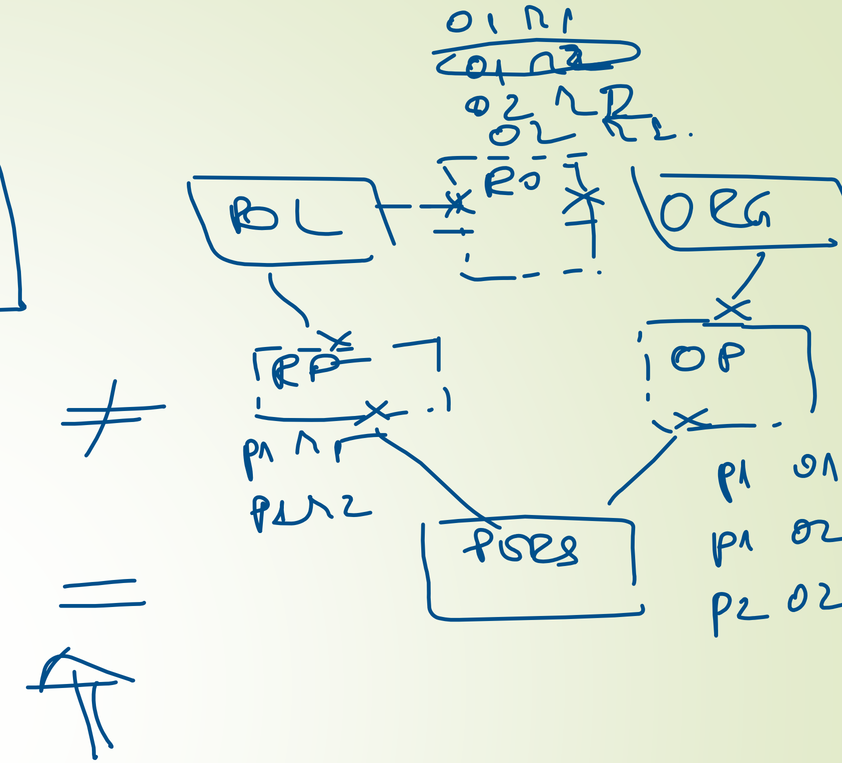
1	P1
1	P2




INFO_PROG (cod-prog#, lbj#, lb_sto#)
1 Java EN
1 SQL FR
1 Java FR
1 SQL EN
2 C++ EN
2 Python DE
2 SQL FR
2 C++ DE
2 C++ FR
cod-prog → lbj =>
cod-prog → lb_sto
INFO_PROG1 (cod_prog#, lbj#)
INFO_PROG2 (cod_prog#, lb_sto#)
R1 = JOIN (Π X U1 (R), Π X U2 (R))
1 Java
1 SQL
2 C++
2 Pyth
2 SQL
1 EN
1 FR
2 EN
2 DE
2 FR

FN5: — Exemplu:





Dependență: OCUPA = join(RO, RP, OP)

⟹ Aducere la FN5: Descompunerea RO, RP, OP.

Regula modelului: Pers. p poate ocupa rolurile $\{r_1, \dots, r_n\}$ și face parte din $\{o_1 - o_p\}$.

Dacă org o_j conține rolurile $\{r_{j1}, \dots, r_{jk}\} \subseteq \{r_1, \dots, r_n\}$ ⟹

⟹ p ocupă fiecare rol r_{jl} în o_j, $l \in \overline{1,k}$, $j \in \overline{1,p}$