3.2 Calcul Relationnel à Variables n-uplets

C'est un langage de la logique des prédicats avec plusieurs *sortes* ou "*types*".
Une variable x aura un type T et on note $x : T$.

Les valeurs des variables sont des n-uplets de la base.

Lien intuitif avec SQL $\rightsquigarrow$

Relation CINEMA, dont deux des attributs sont nom_cinema et adresse, relation SEANCE, dont deux des attributs sont nom_cinema et id_film, et relation FILM dont deux des attributs sont id_film et nom_realisateur.

Quel cinémas ne projettent aucun film réalisé par Tarantino ? Donner aussi les adresses.

```
SELECT c.nom_cinema, c.adresse
FROM  cinema c
WHERE NOT EXISTS (SELECT s.nom_cinema
            FROM seance s, film f
           WHERE f.nom_realisateur = 'Tarantino'
            AND   f.id_film = s.id_film
           AND   s.nom_cinema = c.nom_cinema);
```

Ici, c est une variable dont les valeurs (successives) sont les n-uplets (lignes) de la table SEANCE.
NOT EXISTS (...) code une formule logique qui s'évalue à Vrai ou Faux, selon les valeurs de c.

Deux différences avec la logique des prédicats du cours de logique du L2.

1) La valeur d'une variable n'est pas un individu "indécomposable", mais un n-uplet. Elle a donc des "composantes" :
si $x : ABC$ vaut $\langle a1 : A, b1 : B, c1 : C\rangle$, alors on a une syntaxe permettant d'accéder à la composante $a1$, à la composante $b1$ et à la composante $c1$.
On peut voir le "$.Attribut$" comme un *symbole de fonction* qui associe à un individu, le triplet x,un autre individu du domaine de discours, sa composante $x.Attribut$ (cours de logique du L2).

2) Les variables sont typées.
Utilité ?
Par ex. on a : univers $= \{A, B, C, ..., Z\}$. Parmi les tables, on a r, de schéma $\{A, B, C\}$ (souvent on note juste ABC). On veut exprimer, en logique : tous les n-uplets de r ont une propriété Q.

Si on ne type pas :

$$\forall x(si\ x \in r\ alors\ Q(x))$$

Pour évaluer, on examine **tous** les n-uplets possibles :
avec une seule composante, sur A, avec une seule composante, sur B, $\cdots$ avec une seule composante, sur Z,
avec 2 composantes, sur AB, sur AC,.....,WZ
$\vdots$
avec 26 composantes, sur $AB...Z$

Mais on est intéressé seulement aux n-uplets sur ABC ! Alors :

$$\forall x : ABC(si\ x \in r\ alors\ Q(x))$$

Encore sur les principes du Calcul Relationnel à variables n-uplets (CR à n-uplets).

- Une variable aura comme valeur un n-uplet qui PEUT apparaître dans la base (bon type), mais qui n'y est pas forcement.
- Notation : si x est une variable (n-uplet) et A un attribut, l'expression $x.A$ indique la valeur pour l'attribut A du n-uplet qui est la valeur de x (syntaxe pour accéder aux composantes).

Pour un schéma de base $\mathcal{S}$ donné sur un univers U, le vocabulaire des formules du calcul relationnel est :

- $V =$ ensemble infini de variables.
- $C =$ ensemble de constantes $= \bigcup_{A_i \in U} Dom(A_i)$
- Symboles de prédicat : $=$, $>$, $\geq$, $<$, $\leq$, et tout nom de relation dans $\mathcal{S}$.

On a aussi une fonction *type* qui associe à certaines variables des sous-ensembles de U.
Pour $t \in V$, lire $type(t)$ comme "le type de t".

Syntaxe des **Formules Atomiques** (F.A.) :

1. **Si** :
   $t, t' \in V$, A est un attribut $\in type(t)$ et B est un attribut $\in type(t')$
   alors le mot $t.A = t'.B$ est une F.A.

2. **Si** $t \in V$, A est un attribut $\in type(t)$, $k \in C$ et $k \in Dom(A)$
   alors le mot $t.A = k$ est une F.A.

3. **Si** :
   $t \in V$, R est un nom de relation, S est le schéma de relation associé à R et $type(t) = S$
   alors le mot $R(t)$ est une F.A.

Exemples de F.A..

Soit $Projection$ un nom de relation, dont le schéma est
$\{NomFilm,\ NomCinema,\ Horaire\}$ et soit $Aime$ une relation de schéma
$\{Spectateur, NomFilm\}$.
Soient t et t' des variables, avec
$type(t) = type(t') = \{NomFilm,\ NomCinema,\ Horaire\}$.

Les mots :

1. $t.NomCinema = t'.NomCinema$
2. $t.NomCinema =$ "Gaumont Alésia"
3. $Projection(t)$

sont des F.A.

Le mot $Aime(t)$ n'est pas une F.A. : erreur de typage !

Signification (sémantique) des F.A.

Soit B une base sur le schéma $\mathcal{S}$. Soit n_1 un n-uplet choisi comme valeur de la variable t et soit n_2 un n-uplet choisi comme valeur de la variable t'.

1. $t.A = t'.B$ est vraie par rapport à B ssi la valeur associée à l'attribut A par n_1 est la même que celle associée à l'attribut B par n_2.
2. $t.A = k$ est vraie par rapport à B ssi la valeur associée à l'attribut A par n_1 est k.
3. $R(t)$ est vraie par rapport à B ssi la relation de nom R contient n_1.

N.B. Si $type(t) = E$, où E est un ensemble d'attributs, tout n-uplet sur E peut être une valeur de t, même si n n'appartient à aucune relation !

Exemple. Base B :

$AimeLivre$

Personne	NomLivre
p1	l1
p2	l2
p1	l2

$Livre$

NomLivre
l1
l2

Donnons la valeur $\langle p1, l1\rangle$ à t et $\langle l1\rangle$ à t'.

$t.NomLivre = t'.NomLivre$ est vraie par rapport à B.

$t.NomLivre = l2$ est fausse.

$AimeLivre(t)$ est vraie.

Associons $\langle p2, l1\rangle$ à t et $\langle l2\rangle$ à t'.

$t.NomLivre = t'.NomLivre$ est fausse.

$t.NomLivre = l1$ est vraie.

$AimeLivre(t)$ est fausse.

Formules

$E \subseteq U$=ensemble d'attributs

- Toute F.A. est une formule.
- Si F est une formule alors $(\neg F)$ l'est aussi.
- Si F_1 et F_2 sont des formules et $* \in \{\wedge, \vee, \rightarrow\}$ alors $(F_1 * F_2)$ est une formule.
- Si F est une formule et $t \in V$ alors :
  $(\forall t : E\ F)$ et $(\exists t : E\ F)$
  sont des formules.

Formules : suite

Lecture intuitive de "$\forall t : E\ FF$" : quelque soit le n-uplet t de type E, F est vraie pour t.

Lecture intuitive de "$\exists t : E\ F$" : il existe au moins un n-uplet t de type E tel que F est vraie pour t.

Formules : suite

La même définition de l'ensemble des formules par une grammaire :
$Formule := F.A. \mid (\neg Formule) \mid (Formule * Formule)$
$\mid (\forall t : E\ Formule) \mid (\exists t : E\ Formule)$

où $* \in \{\wedge, \vee, \rightarrow\}$.

Conventions : droit de ne pas écrire les () le plus à l'extérieur. Autres droits : $F_1 \wedge F_2 \wedge F_3$ à la place de $(F_1 \wedge F_2) \wedge F_3$ ou $F_1 \wedge (F_2 \wedge F_3)$; idem pour $\vee$.

Lien avec le cours de logique du L2 : syntaxe du calcul des prédicats (mais types en plus).

Pour $(\forall t : E\ F)$ et $(\exists t : E\ F)$:
F= *portée* de $\forall t$ (resp. $\exists t$).

Variables Libres d'une Formule F : celles qui ne sont pas dans la portée d'un quantificateur ($\forall$ ou $\exists$).

Exemple.
$F : \exists t : Personne, NomLivre$
$(AimeLivre(t) \wedge t.Personne ='' Jean'' \wedge t'.NomLivre = t.NomLivre)$

$$Ensemble\ des\ variables\ libres\ de\ F = \{t'\}$$

F va être vraie pour certain valeurs de t', fausse pour d'autres.

Signification (Sémantique)des formules

F : formule, B : base. s : choix de valeurs pour les variables libres de F.
t : variable n-uplet, $type(t) = E$.
Par rapport à B et s on a :

- Si F est une F.A. , F est vraie ou fausse selon les critères déjà vus.
- Si F est $(\neg F_1)$, F est vraie ssi F_1 est fausse.
- Si F est $(F_1 \wedge F_2)$, F est vraie ssi F_1 est vraie et F_2 aussi.
- Si F est $(F_1 \vee F_2)$, F est vraie si au moins une de F_1, F_2 est vraie.
- Si F est $(F_1 \rightarrow F_2)$, F est vraie ssi ou F_1 est fausse ou F_2 est vraie.
- Si F est $\forall t : E\ F_1$, F est vraie ssi, quelle que soit la valeur de t (ayant le type E), F_1 est vraie pour t.
- Si F est $\exists t : E\ F_1$, F est vraie ssi il existe au moins une valeur de t (ayant le type E) telle que F_1 est vraie pour t.

Lien avec le cours de logique du L2 : ici B est une interprétation (particulière) !

Révenons à la syntaxe et au typage.

Erreurs d'écriture possibles :

Non-Formules :

1. $$(Aime(t) \wedge Livre(t')$$

   Erreur de syntaxe, pas (forcement) de typage.

2. $Aime(t) \wedge t.age = 40$ où $type(t) = \{NomPersonne, NomLivre\}$
   Erreur de typage!

3. (Rappel : le schéma de $Livre$ est $\{NomLivre\}$)

   $$\forall t : Personne \neg Livre(t)$$

   Erreur de typage!

Les exemples qui vont suivre utilisent le schéma de base du premier cours :

$U =$
$\{NomFilm, Realizateur, Acteur, Producteur, NomCinema, Horaire, Spectateur\}$

$$\mathcal{S} =$$

$\{$
$Film(NomFilm, Realizateur, Acteur, Producteur),$
$Projection(NomFilm, NomCinema, Horaire), Aime(Spectateur, NomFilm)$
$\}$

Formulation d'une requête dans le calcul relationnel à variables n-uplets.

- Format :

$$\{t : E \mid F(t)\}$$

où t est une variable, E est le type déclaré pour t et $F(t)$ est une formule ayant t comme seule variable libre.

- Réponse : l'ensemble des des valeurs de t pour lesquelles $F(t)$ est vraie.
- Exemple : "Quels sont les (titres des) films projetés au Gaumont Alésia ?
$\{t : NomFilm \mid \exists t' : NomFilm, NomCinema, Horaire\ (Projection(t')$
$\wedge t'.NomCinema = \text{“}GaumontAlesia'' \wedge t.NomFilm = t'.NomFilm)\}$

Un autre exemple.

"Quels sont les titres et les acteurs des films projetés au Gaumont Alésia ?"

$\{t : NomFilm, Acteur \mid$
$\exists t'' : NomFilm, Acteur, Realisateur, Producteur$
$\exists t' : NomFilm, NomCinema, Horaire$
$(Film(t'') \wedge (Projection(t')$
$\wedge\, t'.NomCinema ='' GaumontAlesia'' \wedge t''.NomFilm = t'.NomFilm \wedge$
$t.NomFilm = t'.NomFilm \wedge t.Acteur = t''.Acteur)\}$

NB : En algèbre on aurait écrit :

$$\pi_{NomFilm,Acteur}\sigma_{NomCinema=``GaumontAlesia''}(Film \bowtie Projection)$$

Question. Même base B, mais domaines des attributs $\neq$.

A-t-on forcement une réponse unique à une requête dans le calcul ? NON.

Exemple : Schéma de la base $=\{R(A)\}$, c.-à-d. on a une seule table, R, dont le schéma est $\{A\}$.

A
1
2
3

Trois Cas

1) $Dom(A) = \{1, 2, 3\}$

2) $Dom(A) = \{1, 2, 3, 4\}$

3) $Dom(A) = \{1, 2, 3, 4, 5, 6, ...\}$

Requête : $\{t : A \mid \neg R(t)\}$

Réponses respectives dans les trois cas : $\emptyset$, $\{4\}$, $\{n \mid n$ est un entier positif et $n > 3\}$.

Dans le troisième cas, la réponse n'est même pas une relation finie ! Pas de correspondance avec l'algèbre relationnelle.

Cette requête est dépendante du domaine. Not Good !

- $\exists$ restriction syntaxique sur la forme d'une requête R du calcul qui assure que R est indépendante du domaine et équivalente à une expression algébrique : notion de requête *saine*.
- Définition de *saine* : techniquement compliquée, pas donnée ici.
- Mais : exemples de réécriture d'une requête dépendante du domaine (D.D.) de façon qu'elle soit indépendante du domaine (I.D.).

Exemples

Schéma de la base : $\{$
$Film(NomFilm, Realizateur, Acteur, Producteur)$,
$Projection(NomFilm, NomCinema, Horaire), Aime(Spectateur, NomFilm)$
$\}$

"Quels films ne sont aimés par personne ?"

D.D. : $\{t : NomFilm \mid \forall x : Spectateur, NomFilm\ (x.NomFilm = t.NomFilm \rightarrow \neg Aime(x))\}$

Evaluer, par exemple, sur $Aime = \{\langle s1, f1\rangle, \langle s2, f2\rangle\}$ avec $Dom(NomFilm) = \{f1, f2, f3, ...\}$.

I.D. : $\{t : NomFilm \mid \exists t' : NomFilm, Realizateur, Acteur, Producteur$
$[Film(t') \wedge$
$\forall x : Spectateur, NomFilm\ (x.NomFilm = t'.NomFilm \rightarrow \neg Aime(x))$
$\wedge\ t'.NomFilm = t.NomFilm]\}$

Exemples, suite

"Quels films ne sont pas projettés au Gaumont-Alésia ?"

D.D. : $\{t : NomFilm \mid \neg\exists t'' : NomFilm, NomCinema, Horaire$
$(Projection(t'') \wedge t''.NomCinema = \text{“}Gau.Al.\text{”} \wedge t''.NomFilm = t.NomFilm)\}$

Evaluer, par exemple, sur $Projection = \{\langle f1, c1, h1\rangle, \langle f2, G.A., h22\rangle\}$ avec $Dom(NomFilm) = \{f1, f2, f3, ...\}$.

I.D. : $\{t : NomFilm \mid \exists t' : NomFilm, Realizateur, Acteur, Producteur$
$[Film(t') \wedge$
$\neg\exists t'' : NomFilm, NomCinema, Horaire\ (Projection(t'') \wedge t''.NomCinema =$
$\text{“}Gau.Al.\text{”} \wedge t''.NomFilm = t'.NomFilm) \wedge t'.NomFilm = t.NomFilm]\}$

Exemples, suite

Encore :

"Quels films ne sont pas projettés au Gaumont-Alésia ?"

Les deux formules suivantes sont I.D. et sont **équivalentes** :

$\{t : NomFilm \mid \exists t' : NomFilm, Realizateur, Acteur, Producteur$
$[Film(t') \wedge$
$\neg \exists t'' : NomFilm, NomCinema, Horaire\ [Projection(t'') \wedge t''.NomCinema =$
$\text{"}Gau.Al.\text{"} \wedge t''.NomFilm = t'.NomFilm] \wedge t'.NomFilm = t.NomFilm)]\}$
(déjà vue).

$\{t : NomFilm \mid \exists t' : NomFilm, Realizateur, Acteur, Producteur$
$[Film(t') \wedge$
$\forall t'' : NomFilm, NomCinema, Horaire\ \neg[Projection(t'') \wedge t''.NomCinema =$
$\text{"}Gau.Al.\text{"} \wedge t''.NomFilm = t'.NomFilm] \wedge t'.NomFilm = t.NomFilm)]\}$

Pourquoi ? $\neg \exists x F(x) \equiv \forall x \neg F(x)$.

Et encore une autre écriture équivalente si on remarque que $\neg(A \wedge B \wedge C) \equiv ((A \wedge B) \rightarrow \neg C)$.

- algèbre relationnelle et calcul relationnel à variables n-uplets (requêtes I.D.) : expression des mêmes requêtes.
- Un langage commercial est dit *relationnellement complet* ssi il peut exprimer toutes les requêtes que l'algèbre peut exprimer.
- SQL est relationnellement complet.
- SQL utilise des opérateurs de l'algèbre, mais aussi des variables n-uplets.