

# Introduction aux Bases de données

## **Cours 2 :** Calcul Relationnel des tuples

LU2IN009

# Interrogation des données

## Algèbre relationnelle (voir 3I009)

- Langage procédural : comment calculer les données
- Fondé sur la théorie des ensembles

## Calcul relationnel des n-uplets

- Langage déclaratif : qu'est ce qu'on veut calculer
- Fondé sur la logique des prédicats

## SQL (Structured Query Language)

- Langage basé sur le calcul relationnel
- Comporte en plus des fonctions d'agrégation
- Implante également les opérateurs de l'algèbre

# Rappel logique 1er ordre

- Appelée également logique des prédicats
- Système formel utilisé en Maths, Philo, Info.
- Enrichit la logique propositionnelle avec notion de variables et de quantificateurs

## Logique propositionnelle

si Jean est malade  
alors il ne sort pas  
Jean est malade  
**conclusion : Jean ne sort pas**

si  $p$  alors non  $q$

$p$

---

**non  $q$**

## Logique des prédicats

si un homme est malade  
alors il ne sort pas  
**Jean est malade**  
**conclusion : Jean ne sort pas**

si  $P(x)$  alors non  $Q(x)$

$P(\text{Jean})$

---

**non  $Q(\text{Jean})$**

# Calcul Relationnel des n-uplets :

## aperçu

Langage fondé sur la logique des prédicats

- Tuple avec n attributs → prédicat n-aire
- Valeur atomique → constante

**Etudiants**

**Tuples**

<u>matricule</u>	nom	prénom	dateNaiss	adresse
1753	Smith	Joe	1992-01-12	11 CP NYC
9832	Smith	Dan	1989-04-03	22 Rd NJ
....				

**Prédicats**

Etudiants('1753','smith', 'Joe', '1992-01-12', '11 CP NYC')  
Etudiants('9832','smith', 'Dan', '1989-04-03', '22 Rd NJ')  
...

# Calcul Relationnel des n-uplets : aperçu

Langage fondé sur la logique des prédicats

- Formule logique pour exprimer une condition devant être respectée par les n-uplets à retourner

Requête : retourner les étudiants ayant pour nom 'Smith'

Condition : **Etudiants(x) " x.nom='Smith'**

**Etudiants**

<u>matricule</u>	nom	prénom	dateNaiss	adresse
1753	Smith	Joe	1992-01-12	11 CP NYC
9832	Smith	Dan	1989-04-03	22 Rd NJ
....				

**Tuples**

**Prédicats**

Etudiants('1753','smith', 'Joe', '1992-01-12', '11 CP NYC')

Etudiants('9832','smith', 'Dan', '1989-04-03', '22 Rd NJ')

...

# Calcul Relationnel des n-uplets : aperçu

Langage fondé sur la logique des prédicats

- Formule logique pour exprimer une condition devant être respectée par les n-uplets à retourner

Requête : retourner les étudiants ayant pour nom 'Smith'

Condition : **Etudiants(x) " x.nom='Smith'**

**Etudiants**

Réponse

Tuples

<u>matricule</u>	nom	prénom	dateNaiss	adresse
1753	<b>Smith</b>	Joe	1992-01-12	11 CP NYC
9832	<b>Smith</b>	Dan	1989-04-03	22 Rd NJ
....				

Prédicats

Etudiants('1753','smith', 'Joe', '1992-01-12', '11 CP NYC')

Etudiants('9832','smith', 'Dan', '1989-04-03', '22 Rd NJ')

...

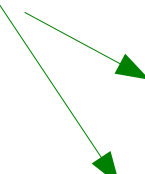
# Calcul Relationnel des n-uplets : aperçu

Requête : retourner les étudiants inscrits en 2I009

Condition : **Etudiants(x)** " **Inscriptions(y)** " **x.matricule=y.matricule**  
" **y.code='2I009'**

Réponse

Etudiants



<u>matricule</u>	nom	prénom	dateNaiss	adresse
1753	<b>Smith</b>	Joe	1992-01-12	11 CP NYC
9832	<b>Smith</b>	Dan	1989-04-03	22 Rd NJ
....				

Inscriptions

<u>Matricule*</u>	<u>Code*</u>
1753	<b>2I009</b>
1753	2I002
9832	<b>2I009</b>

# Calcul Relationnel des n-uplets : aperçu

**Requête :** retourner les étudiants inscrits en Bases de Données ('BD')

*Condition :* **Etudiants(x)** " **Inscriptions(y)** " **x.matricule=y.matricule** " **Modules(z)** " **y.code=z.code** " **z.intitule='BD'**

**Etudiants**

<u>matricule</u>	nom	prénom	...
1753	Smith	Joe	
9832	Smith	Dan	
....			

**Inscriptions**

<u>Matricule*</u>	<u>Code*</u>
1753	LI341
1753	LI345
9832	LI341

**Modules**

<u>code</u>	intitulé	niveau
LI341	BD	L3
LI345	Web	L3
LI399	Crypto	L3

Autres requêtes :

- retourner les étudiants inscrits dans un même module que 'Jack'?
- retourner les étudiants inscrits dans aucun module?
- retourner les étudiants inscrits dans tous les modules?
- ....



# Calcul Relationnel des n-uplets :

## syntaxe

Forme générale des requêtes

{ **expression** | **Condition** }



Format résultat



Formule logique du 1er ordre

Expression	Signification
$v, w, \dots z$	tous les attributs plusieurs tables
$v.a1, v.a2, \dots v.am$	certain attributs même table
$v.a1..v.am, w.b1..w.b$ $p ..$	certain attributs plusieurs tables

# Calcul Relationnel des n-uplets : syntaxe

Forme générale des requêtes

{ **expression** | **Condition** }



Format résultat



Formule logique du 1er ordre

- **$R(v)$**
- **$v.A \text{ op } w.B$**  où  $\text{opa} \{=, \neq, >, <, \leq, \geq\}$
- **$v.A \text{ op 'val'}$**  valeur atomique
- **$C1 \wedge C2, C1 \vee C2, \neg C, C1 \rightarrow C2$**
- **$\prod_v (C), \sum_w (C)$**   $v$  et  $w$  variables liées

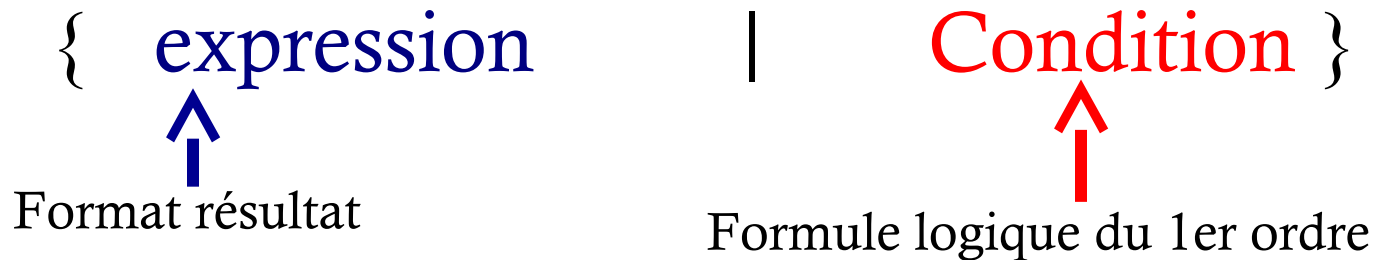
Les variables libres apparaissent dans *expression* et *condition*, les variables liées uniquement dans *condition* {  $\text{expression}(v_1, v_1, \dots, v_n) \mid \text{condition}(v_1, v_1, \dots, v_n, v_{n+1}, \dots, v_m)$  }

# Calcul Relationnel des n-uplets : syntaxe

- Priorité des connecteurs logiques
  - Négation  $\neg$
  - Conjonction  $\wedge$
  - Disjonction  $\vee$
  - Implication  $\rightarrow$
- Utilisation des parenthèses pour lever l'ambiguïté

# Calcul Relationnel des n-uplets : sémantique

Forme générale des requêtes



Retourner les n-uplets spécifiés par *expression* tel  
que *Condition* est vérifiée

# Calcul Relationnel des n-uplets : sémantique

Forme condition	Nom	Sémantique
<b><math>R(v)</math> ou <math>va R</math></b>	Liaison de variables	Vraie lorsque la table <b><math>R</math></b> contient des n-uplets
<b>1. <math>v.A \text{ op } w.B</math></b> <b>2. <math>v.A \text{ op 'val'}</math></b>	Expressions de comparaisons	1. Vraie lorsque l'attribut <b><math>A</math></b> du n-uplet <b><math>v</math></b> est égal/différent de/...l'attribut <b><math>B</math></b> du n-uplet <b><math>w</math></b> 2. Vraie lorsque l'attribut <b><math>A</math></b> du n-uplet <b><math>v</math></b> est égal/différent de/... <b><math>val</math></b>
<b>1. <math>C1 \wedge C2</math></b> <b>2. <math>C1 \vee C2</math></b> <b>3. <math>\neg C</math></b>	Expressions booléennes	1. Vraie lorsque les deux conditions vraies 2. Vraie lorsque l'une des deux conditions vraie 3. Vraie lorsque <b><math>C</math></b> est fausse
<b><math>\exists v (C)</math></b>	Quantificateur existentiel	vrai si <b><math>C</math></b> est vrai pour au moins un n-uplet <b><math>v</math></b>
<b><math>\sum w (C)</math></b>	Quantificateur universel	vrai si <b><math>C</math></b> est vrai pour tout les n-uplets <b><math>w</math></b>

# Calcul Relationnel des n-uplets : exemples

**Etudiants**(matricule, nom, prenom, adresse, collaborateur\*)

**Inscriptions**(matricule\*, code\*)

**Modules**(code, intitule, niveau, salle\*)

**Salles**(numero, capacite, précédent \*, suivante\*)

1. Tous les étudiants
2. Le prénom des étudiants ayant pour nom de famille Smith
3. Le matricule des étudiants inscrits dans le module 'BD' ainsi que le code de ce module
4.
  - a) Les intitulés des modules où 'Jack' est inscrit
  - b) Les étudiants inscrits dans un même module que 'Jack' ainsi que le code de ce module

# Sélection

**Etudiants**(matricule, nom, prenom, adresse, collaborateur\*)

**Inscriptions**(matricule\*, code\*)

**Modules**(code, intitule, niveau, salle\*)

**Salles**(numero, capacite, précédent \*, suivante\*)

- Requête 1: Quels sont tous les étudiants ?

# Sélection

**Etudiants**(matricule, nom, prenom, adresse, collaborateur\*)

**Inscriptions**(matricule\*, code\*)

**Modules**(code, intitule, niveau, salle\*)

**Salles**(numero, capacite, précédent \*, suivante\*)

- Requête 2: Le prénom des étudiants ayant pour nom de famille Smith ?



# Jointure

**Etudiants**(matricule, nom, prenom, adresse, collaborateur\*)

**Inscriptions**(matricule\*, code\*)

**Modules**(code, intitule, niveau, salle\*)

**Salles**(numero, capacite, précédent \*, suivante\*)

- Requête 3: Le matricule des étudiants inscrits dans le module 'BD' ainsi que le code de ce module ?

# Illustration évaluation requête 3

**Etudiants**

e →

<u>matricule</u>	nom	prénom	...
1753	Smith	Joe	
9832	Smith	Dan	
....			

**Inscriptions**

i →

<u>Matricule*</u>	<u>Code*</u>
1753	LI341
1753	LI345
9832	LI341

**Modules**

m →

<u>code</u>	intitulé	niveau
LI341	BD	L3
LI345	Web	L3
LI399	Crypto	L3

# Jointure

**Etudiants**(matricule, nom, prenom, adresse, collaborateur\*)

**Inscriptions**(matricule\*, code\*)

**Modules**(code, intitule, niveau, salle\*)

**Salles**(numero, capacite, précédent \*, suivante\*)

- Requête 4-a) Les intitulés des modules où 'Jack' est inscrit ?

# Evaluation requête 4-a

**Etudiants**

e →

<u>matricule</u>	nom	prénom	...
1753	Smith	Joe	
9832	Smith	Jack	
....			

**Inscriptions**

i →

<u>Matricule*</u>	<u>Code*</u>
1753	LI341
1753	LI345
9832	LI341

**Modules**

m →

<u>code</u>	intitulé	niveau
LI341	BD	L3
LI345	Web	L3
LI399	Crypto	L3

# Jointure

**Etudiants**(matricule, nom, prenom, adresse, collaborateur\*)

**Inscriptions**(matricule\*, code\*)

**Modules**(code, intitule, niveau, salle\*)

**Salles**(numero, capacite, précédent \*, suivante\*)

- Requête 4-b) Les étudiants inscrits dans un même module que 'Jack' ainsi que l'intitulé de ces modules ?

# Evaluation requête 4-b

Etudiants

	<u>matricule</u>	nom	prénom	...
e →	1753	Smith	Joe	
j →	9832	Smith	Jack	
	....			

Inscriptions

	<u>Matricule*</u>	<u>Code*</u>
i1 →	1753	LI341
	1753	LI345
i2 →	9832	LI341

Modules

	<u>code</u>	intitulé	niveau
m →	LI341	BD	L3
	LI345	Web	L3
	LI399	Crypto	L3

# Différence

**Etudiants**(matricule, nom, prenom, adresse, collaborateur\*)

**Inscriptions**(matricule\*, code\*)

**Modules**(code, intitule, niveau, salle\*)

**Salles**(numero, capacite, précédent \*, suivante\*)

- Requête 5-a) Les étudiants qui ne sont pas inscrits au module 'LI345'?

# Evaluation requête 5-a)

**Etudiants**

e →

<u>matricule</u>	nom	prénom	...
1753	Smith	Joe	
9832	Smith	Jack	
....			

**Inscriptions**

i →

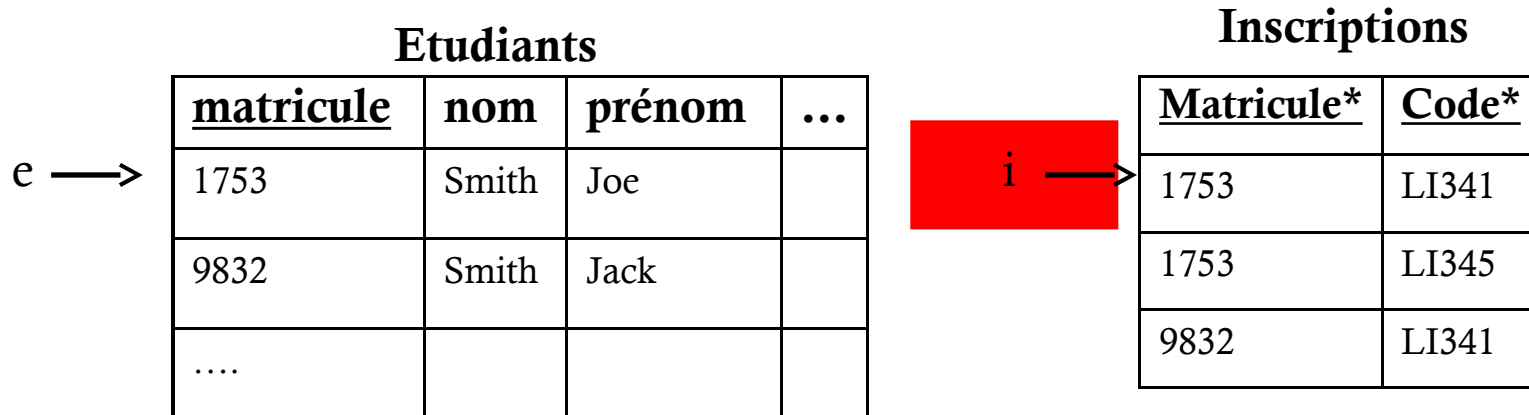
<u>Matricule*</u>	<u>Code*</u>
1753	LI341
1753	LI345
9832	LI341

$Q(e) = \{e.\text{matricule} \mid e \in \text{Etudiants} \wedge \exists i \in \text{Inscriptions} \\ (i.\text{matricule} = e.\text{matricule} \wedge i.\text{code} \neq \text{'LI345'}) \}$

Que contient  $Q(e)$  ? Pourquoi ?



# Evaluation requête 5-a)



$$Q(e) = \{e.\text{matricule} \mid e \in \text{Etudiants} \wedge \neg (\exists i \in \text{Inscriptions} (i.\text{matricule} = e.\text{matricule} \wedge i.\text{code} \neq \text{'LI345'}))\}$$

Que contient  $Q(e)$  ? Pourquoi ?

$$Q(e) = \{e.\text{matricule} \mid e \in \text{Etudiants} \wedge \neg (\exists i \in \text{Inscriptions} (i.\text{matricule} = e.\text{matricule} \wedge i.\text{code} = \text{'LI345'}))\}$$

# Différence

**Etudiants**(matricule, nom, prenom, adresse, collaborateur\*)

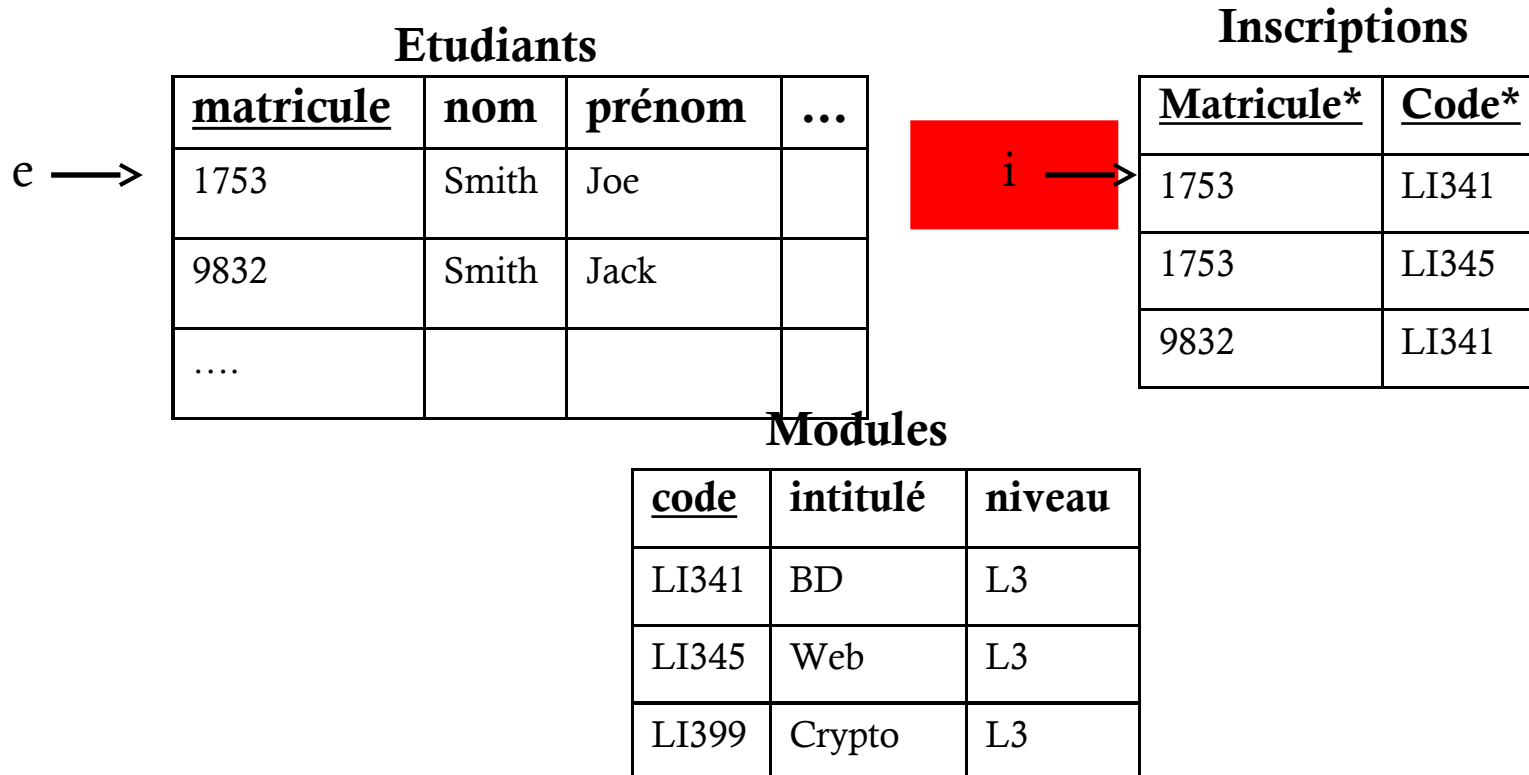
**Inscriptions**(matricule\*, code\*)

**Modules**(code, intitule, niveau, salle\*)

**Salles**(numero, capacite, précédent \*, suivante\*)

- Requête 5-b) Les étudiants qui ne sont inscrits dans aucun module?

# Evaluation requête 5-b)



# Division

**Etudiants**(matricule, nom, prenom, adresse, collaborateur\*)

**Inscriptions**(matricule\*, code\*)

**Modules**(code, intitule, niveau, salle\*)

**Salles**(numero, capacite, précédent \*, suivante\*)

- Requête 6: Les étudiants inscrits dans tous les modules?

# Evaluation requête 6

**Etudiants**

e →

<u>matricule</u>	nom	prénom	...
1753	Smith	Joe	
9832	Smith	Jack	
....			

**Inscriptions**

i →

<u>Matricule*</u>	<u>Code*</u>
1753	LI341
1753	LI345
9832	LI341

**Modules**

m →

<u>code</u>	intitulé	niveau
LI341	BD	L3
LI345	Web	L3

# Jointures

**Etudiants**(matricule, nom, prenom, adresse, collaborateur\*)

**Inscriptions**(matricule\*, code\*)

**Modules**(code, intitule, niveau, salle\*)

**Salles**(numero, capacite, précédent \*, suivante\*)

- Requête 7: Les étudiants inscrits dans au moins deux modules ?

# Evaluation requête 7

**Etudiants**

e →

<u>matricule</u>	nom	prénom	...
1753	Smith	Joe	
9832	Smith	Jack	
....			

**Inscriptions**

i1 →

i2 →

<u>Matricule*</u>	<u>Code*</u>
1753	LI341
1753	LI345
9832	LI341

**Modules**

<u>code</u>	intitulé	niveau
LI341	BD	L3
LI345	Web	L3
LI399	Crypto	L3

# Jointure

**Etudiants**(matricule, nom, prenom, adresse, collaborateur\*)

**Inscriptions**(matricule\*, code\*)

**Modules**(code, intitule, niveau, salle\*)

**Salles**(numero, capacite, précédent \*, suivante\*)

- Requête 8: Les étudiants inscrits à au moins deux modules de niveau L3?



# Evaluation requête 8

**Etudiants**

e →

<u>matricule</u>	nom	prénom	...
1753	Smith	Joe	
9832	Smith	Jack	
....			

**Modules**

m1 →

m2 →

<u>code</u>	intitulé	niveau
LI341	BD	L3
LI345	Web	L3
LI399	Crypto	L3
M009	BD avancées	M1

**Inscriptions**

i1 →

i2 →

<u>Matricule*</u>	<u>Code*</u>
1753	LI341
1753	M009
9832	LI341
1753	LI345

# Jointure

**Etudiants**(matricule, nom, prenom, adresse, collaborateur\*)

**Inscriptions**(matricule\*, code\*)

**Modules**(code, intitule, niveau, salle\*)

**Salles**(numero, capacite, précédent \*, suivante\*)

- Requête 9-a): Les étudiants inscrits à au moins deux modules de niveau L3 ou de niveau M1 ?

# Evaluation requête 9-a)

**Etudiants**

e →

<u>matricule</u>	nom	prénom	...
1753	Smith	Joe	
9832	Smith	Jack	
....			

**Modules**

m1 →

m2 →

<u>code</u>	intitulé	niveau
LI341	BD	L3
LI345	Web	L3
LI399	Crypto	L3
M009	BD avancées	M1

**Inscriptions**

i1 →

i2 →

<u>Matricule*</u>	<u>Code*</u>
1753	LI341
1753	M009
9832	LI341
1753	LI345

# Jointure

**Etudiants**(matricule, nom, prenom, adresse, collaborateur\*)

**Inscriptions**(matricule\*, code\*)

**Modules**(code, intitule, niveau, salle\*)

**Salles**(numero, capacite, précédent \*, suivante\*)

- Requête 9-b) Les étudiants qui sont soit inscrits à au moins deux modules de niveau L3 soit à un module de niveau M1?

# Evaluation requête 9-b)

**Etudiants**

e →	<u>matricule</u>	<b>nom</b>	<b>prénom</b>	<b>...</b>
	1753	Smith	Joe	
	9832	Smith	Jack	
	....			

**Modules**

	<u>code</u>	<b>intitulé</b>	<b>niveau</b>
m1 →	LI341	BD	L3
m2 →	LI345	Web	L3
	LI399	Crypto	L3
m →	M009	BD avancées	M1

**Inscriptions**

	<u>Matricule*</u>	<u>Code*</u>
i1 →	1753	LI341
i →	1753	M009
	9832	LI341
i2 →	1753	LI345

# Cas particuliers

## ● Requêtes sans réponse

- $\{ e \mid \text{Etudiants}(e) \text{ `` Modules } (e) \}$  : deux tables ne peuvent avoir exactement le même n-uplet (schéma différents)
- $\{ e \mid \text{Etudiants}(e) \text{ `` } e.\text{salaire} > 100 \}$  : salaire n'est pas attribut de Etudiants

## ● Requêtes avec réponse infinie $\rightarrow$ requête pas sûre

- $\{ e \mid \neg \text{Etudiants}(e) \}$  : chercher partout sauf dans la table Etudiants
- $\{ e \mid \forall x \text{ Etudiants}(x) \text{ `` } \dots \}$  : tous les n-uplets du monde doivent être dans Etudiant
- $\{ e \mid e.\text{att} > 5 \}$  : e instanciée un nombre infini car pas liée à une table

Les requêtes sur une BD **doivent** être sûres !

# Requêtes sûres

- Bonnes pratiques

- Avec  $\forall$ , il faut toujours un  $\Rightarrow$  qui suit

- Forme équivalente qu'on retrouve dans SQL:

$$\{ t \mid \text{Table}(t) \dots \bigwedge \forall \square \text{TableBis} \dots \neg \bigwedge \forall \square \text{TableTer} \dots \}$$
$$\forall g (p1 \Rightarrow p2)$$
$$\neg (\bigwedge g \neg (p1 \Rightarrow p2))$$
$$\neg g (\bigwedge g \neg (\neg p1 \vee p2))$$
$$\neg g (\bigwedge g (p1 \wedge \neg p2))$$

# Conclusion

- Présentation d'un langage de requête fondé sur la Logique du Premier Ordre
  - Les prédicats sont des tables, les constantes sont les valeurs atomiques, les variables sont liées aux n-uplets
  - Exprimer une requête = spécifier une condition logique  
→ langage déclaratif
  - Autre variante : les variables liées aux attributs (calcul du domaine)
  - Avantage calcul des n-uplets : traduction quasi-directe vers SQL
- Prochain cours : SQL