

Introduction aux Bases de données

Cours 3 : 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 prédictats
- 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 prédictats

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(Jean)

non Q(Jean)

Calcul Relationnel des n-uplets :

aperçu

Langage fondé sur la logique des prédictats

- 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 prédicts

- 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) \wedge x.nom=‘Smith’**

Tuples

Etudiants				
<u>matricule</u>	<u>nom</u>	<u>prénom</u>	<u>dateNaiss</u>	<u>adresse</u>
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 prédicts

- 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) \wedge x.nom=‘Smith’**

Etudiants					
Réponse	matricule	nom	prénom	dateNaiss	adresse
Tuples	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

Requête : retourner les étudiants inscrits en 2I009

Condition : $\text{Etudiants}(x) \wedge \text{Inscriptions}(y) \wedge x.\text{matricule} = y.\text{matricule}$
 $\wedge y.\text{code} = '2I009'$

Réponse

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

Inscriptions

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

Calcul Relationnel des n-uplets : aperçu

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

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

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

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

<u>code</u>	<u>intitulé</u>	<u>niveau</u>
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

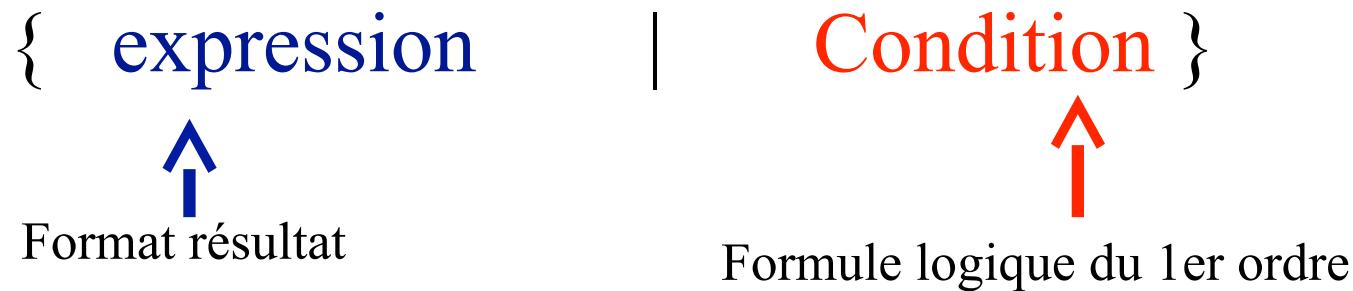


Formule logique du 1er ordre

Expression	Signification
v, w, ...z	tous les attributs plusieurs tables
v.a1, v.a2, ...v.am	certains attributs même table
v.a1..v.am,w.b1..w.bp ..	certains attributs plusieurs tables

Calcul Relationnel des n-uplets : syntaxe

Forme générale des requêtes



- **R(v)**
- **v.A op w.B où $\text{op} \in \{=, \neq, >, <, \leq, \geq\}$**
- **v.A op ‘val’ valeur atomique**
- **C1 \wedge C2 , C1 \vee C2, \neg C, C1 \Rightarrow C2**
- **$\exists v (C) , \forall w (C)$ v et w variables liées**

Les variables libres apparaissent dans *expression* et *condition*, les variables liées uniquement dans *condition* { expression(v1,v1,...,vn) | condition(v1,v1,...,vn, vn+1, ..., vm)}

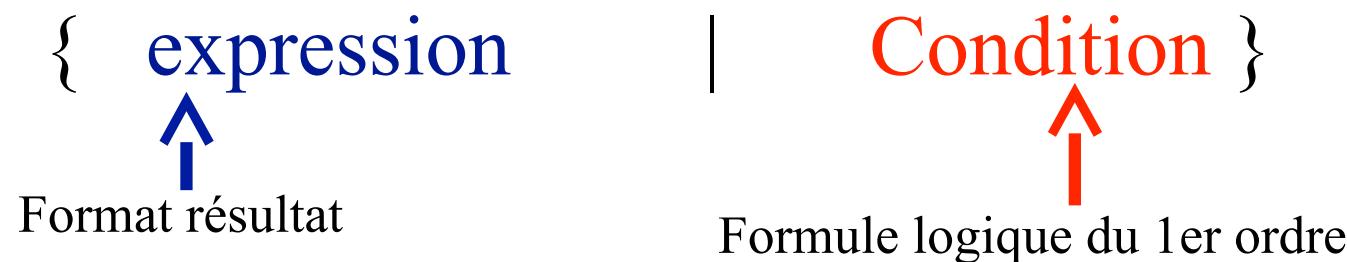
Calcul Relationnel des n-uplets :

syntaxe

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

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
$R(v) \text{ ou } v \in R$	Liaison de variables	Vraie lorsque la table R contient des n-uplets
$1. \ v.A \text{ op } w.B$ $2. \ v.A \text{ op } 'val'$	Expressions de comparaisons	<ol style="list-style-type: none"> 1. Vraie lorsque l'attribut A du n-uplet v est égal/différent de... l'attribut B du n-uplet w 2. Vraie lorsque l'attribut A du n-uplet v est égal/différent de... val
$1. \ C1 \wedge C2$ $2. \ C1 \vee C2$ $3. \ \neg C$	Expressions booléennes	<ol style="list-style-type: none"> 1. Vraie lorsque les deux conditions vraies 2. Vraie lorsque l'une des deux conditions vraie 3. Vraie lorsque C est fausse
$\exists v (C)$	Quantificateur existentiel	vrai si C est vrai pour au moins un n-uplet v
$\forall w (C)$	Quantificateur universel	vrai si C est vrai pour tout les n-uplets w

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*)

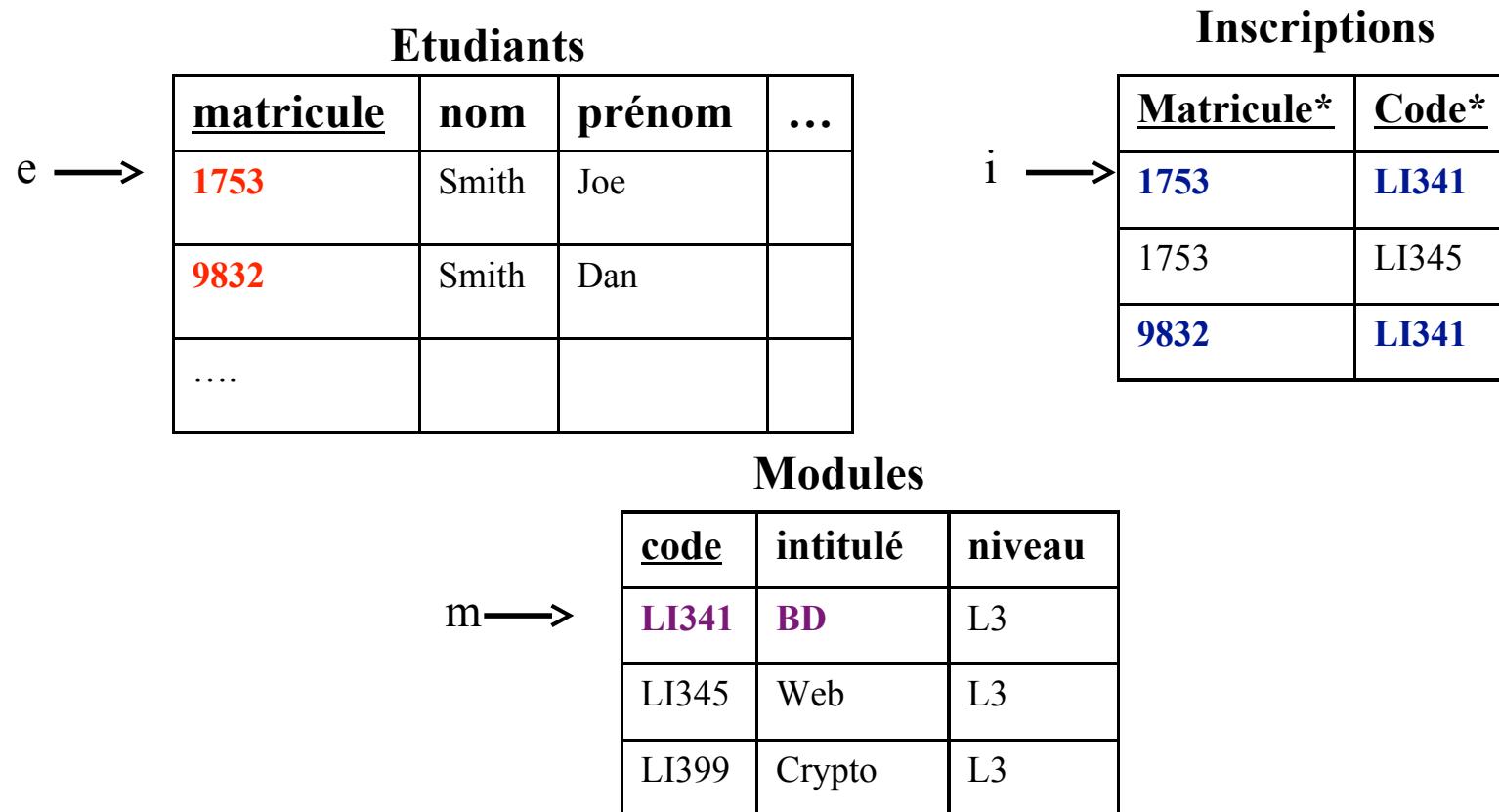
Inscriptions(matricule*, code*)

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



Jointure

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

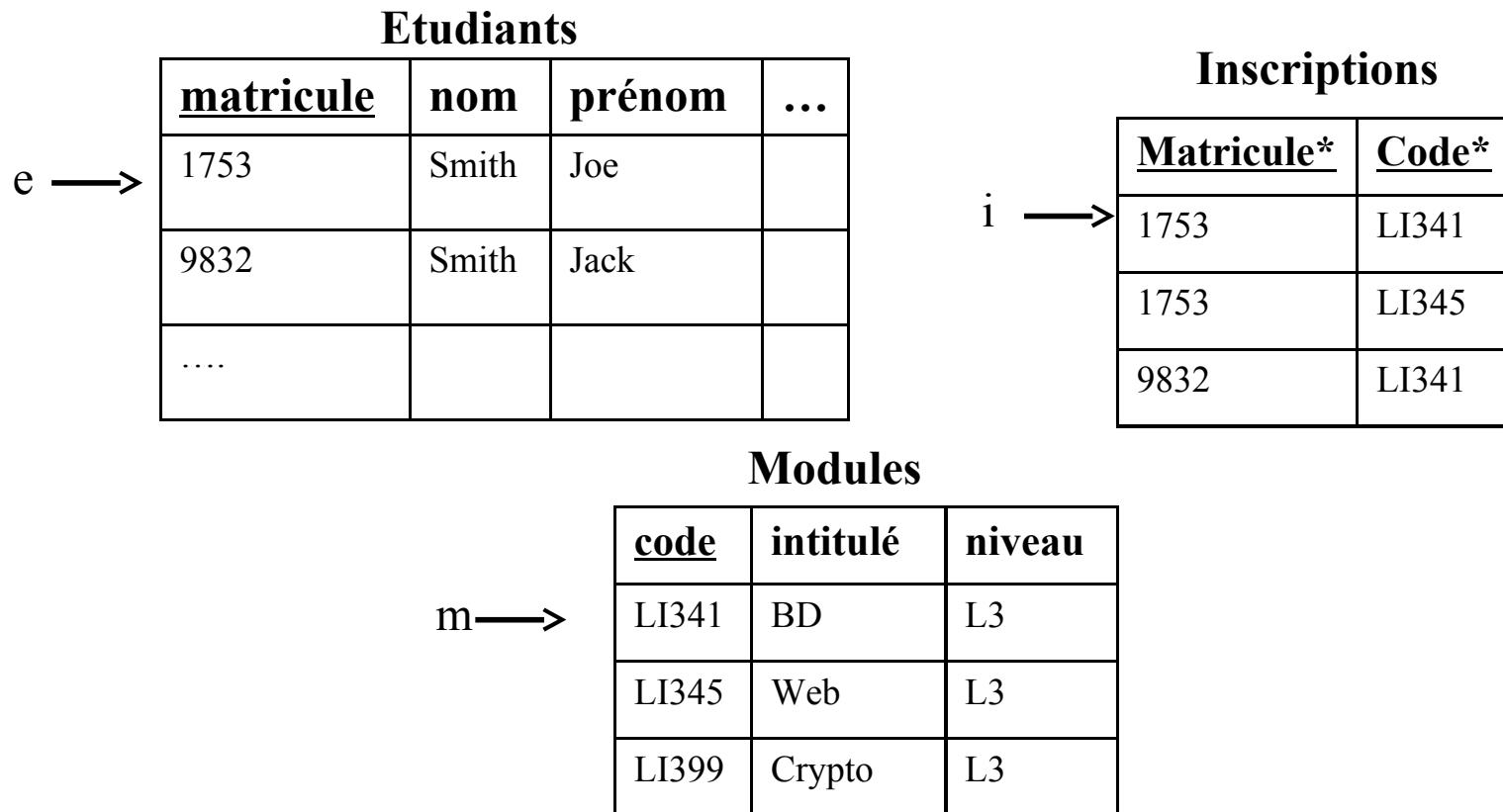
Inscriptions(matricule*, code*)

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



Jointure

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

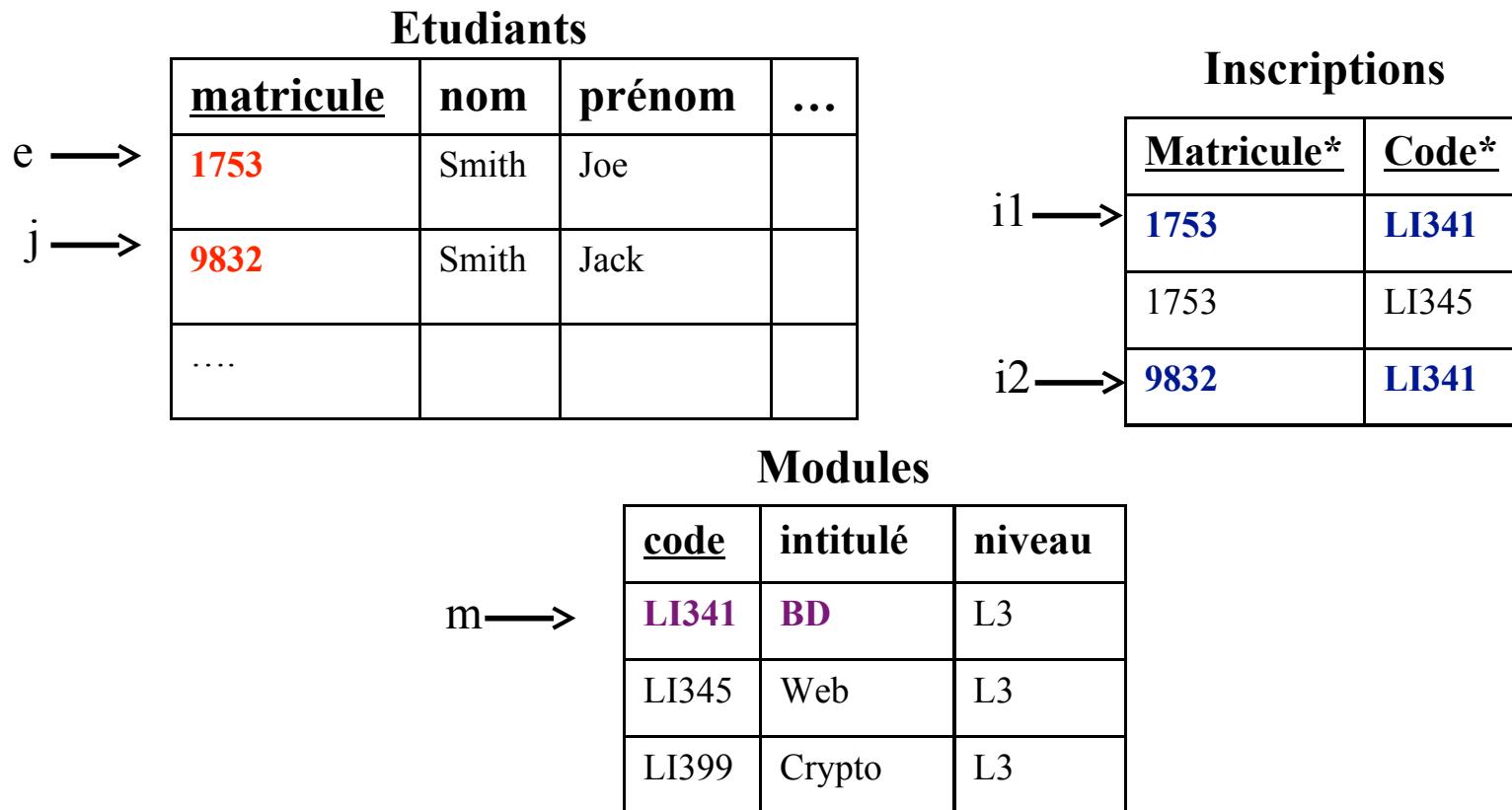
Inscriptions(matricule*, code*)

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



Différence

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

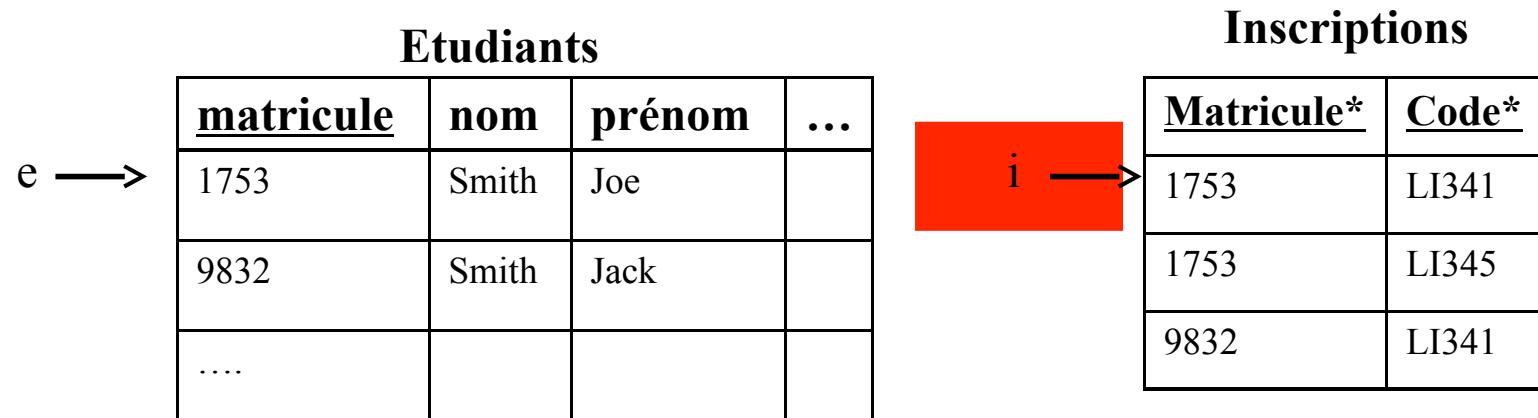
Inscriptions(matricule*, code*)

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)



Que contient $Q(e)$? Pourquoi ?

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?

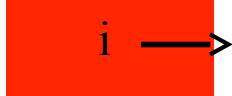
Evaluation requête 5-b)

e →

Etudiants

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

Inscriptions



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

Modules

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

Division

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

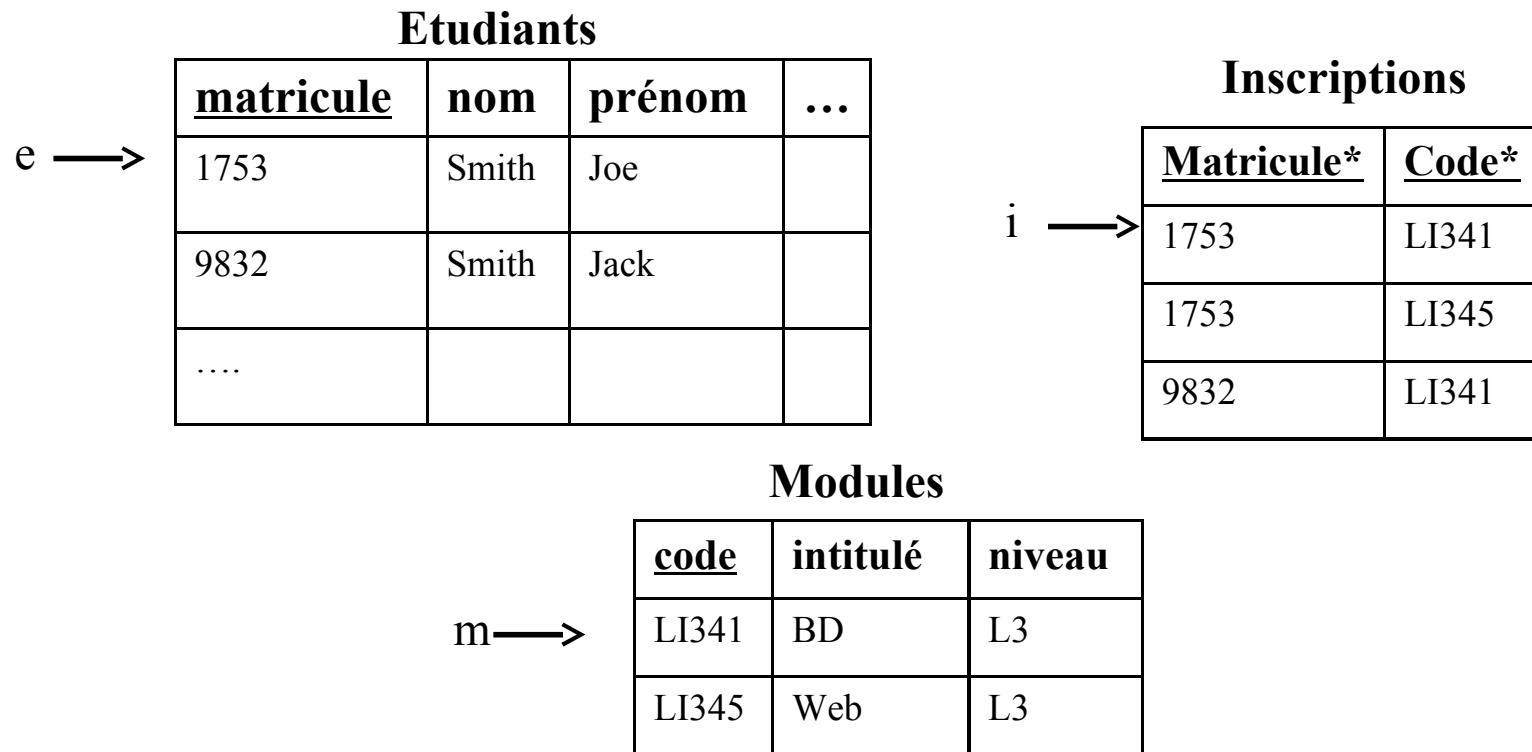
Inscriptions(matricule*, code*)

Modules(code, intitule, niveau, salle*)

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

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

Evaluation requête 6



Jointures

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

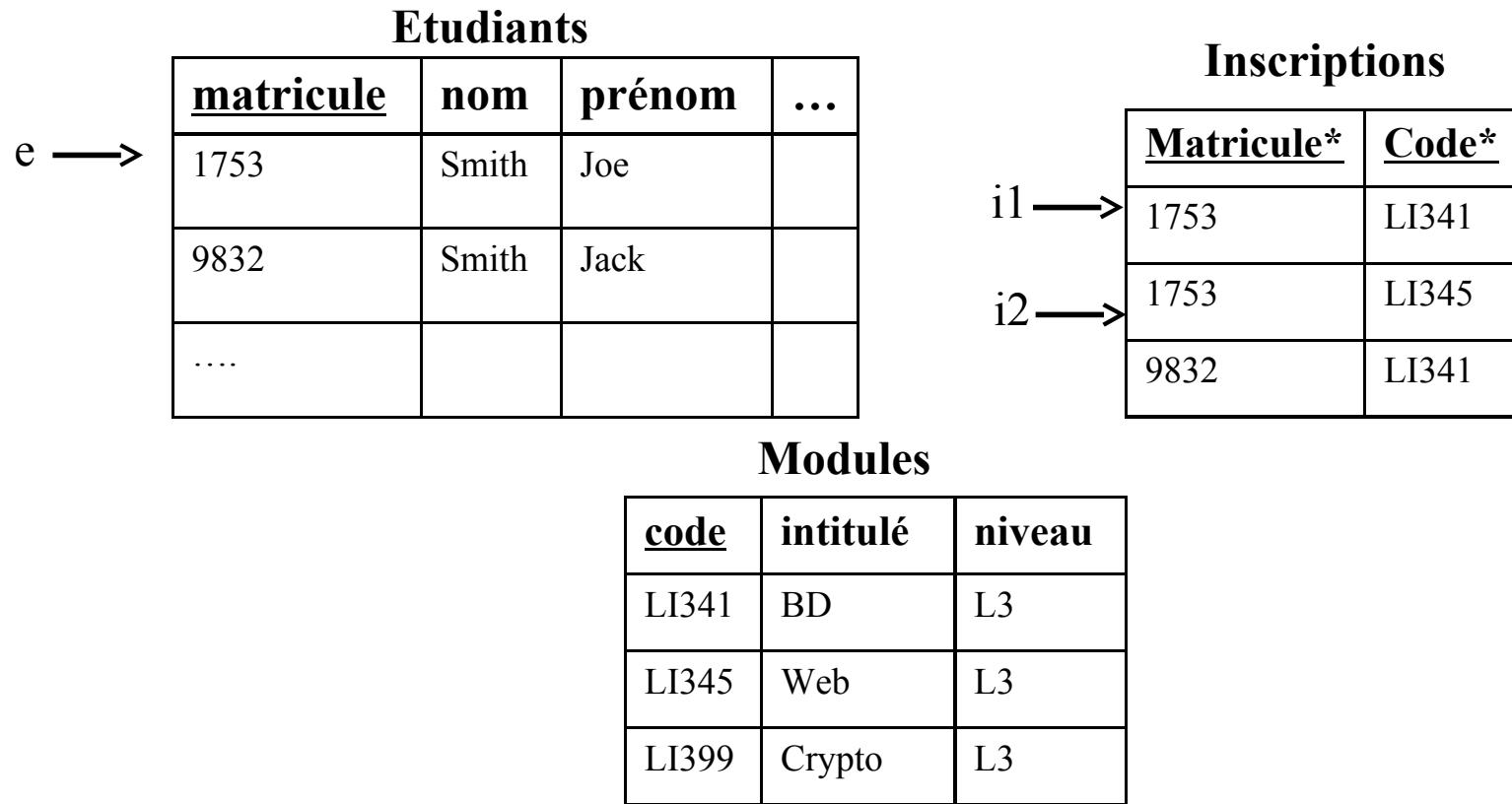
Inscriptions(matricule*, code*)

Modules(code, intitule, niveau, salle*)

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

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

Evaluation requête 7



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

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

Modules

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

e →

Inscriptions

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

i1 →

m1 →

m2 →

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

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

Modules

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

e →

m1 →

m2 →

Inscriptions

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

i1 →

i2 →

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

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

Modules

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

e →

m1 →

m2 →

m →

Inscriptions

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

Cas particuliers

● Requêtes sans réponse

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

● Requêtes avec réponse infinie → requête pas sûre

- $\{ e \mid \neg \text{Etudiants}(e) \}$: chercher partout sauf dans la table Etudiants
- $\{ e \mid \forall x \text{ Etudiants}(x) \wedge \dots \}$: 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:
 $\{ t \mid \text{Table}(t) \dots \exists v \in \text{TableBis} \dots \neg \exists v \in \text{TableTer} \dots \}$

$$\forall g (p1 \Rightarrow p2)$$

$$\neg(\exists g \neg(p1 \Rightarrow p2))$$

$$\neg g(\exists g \neg(\neg p1 \vee p2))$$

$$\neg g (\exists g (p1 \wedge \neg p2))$$

Conclusion

- Présentation d'un langage de requête fondé sur la Logique du Premier Ordre
 - Les prédictats 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