Introduction aux Bases de données
Cours 3 : Calcul Relationnel

UFR 919 – Licence

2e année

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

- appelé 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(Jean)
_____
non Q(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

matricule	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

matricule	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

matricule	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

Requête : retourner les étudiants inscrits en LI341
Condition : Etudiants(x) ∧ Inscriptions(y) ∧x.matricule=y.matricule ∧ y.code='LI341'

Etudiants

Réponse

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

Inscriptions

Matricule*	Code*
1753	LI341
1753	LI345
9832	LI341

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

matricule	nom	prénom	…
1753	Smith	Joe	
9832	Smith	Dan	
….			

Inscriptions

Matricule*	Code*
1753	LI341
1753	LI345
9832	LI341

Modules

code	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, …z	tous les attributs plusieurs tables
$v.a_1$, $v.a_2$, …$v.a_m$	certains attributs même table
$v.a_1..v.a_m, w.b_1..w.b_p$ ..	certains 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 op w.B où op∈{=,≠,>,<,≤,≥}**
- **v.A op 'val' valeur atomique**
- **$C_1 \wedge C_2$, $C_1 \vee C_2$, ¬C, C1➔C2**
- **∃v (C) , ∀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 ¬
  - Conjonction ∧
  - Disjonction ∨
  - Implication ➔
- Utilisation des parenthèses pour lever l'ambiguité

Calcul Relationnel des n-uplets : sémantique

Forme générale des requêtes

{ expression | Condition }

Format résultat — Formule logique du 1er ordre

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

Calcul Relationnel des n-uplets : sémantique

Forme condition	Nom	Sémantique
R(v) ou v∈R	Liaison de variables	Vraie lorsque la table **R** contient des n-uplets
1. v.A op w.B **2. v.A op 'val'**	Expressions de comparaisons	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. $C_1 \wedge C_2$ **2. $C_1 \vee C_2$** **3. ¬C**	Expressions booléennes	1. Vraie lorsque les deux conditions vraies 2. Vraie lorsque l'une des deux conditions vraie 3. Vraie lorsque **C** est fausse
∃v (C)	Quantificateur existentiel	vrai si **C** est vrai pour au moins un n-uplet **v**
∀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*)

1. Tous les étudiants

Sélection

Etudiants(matricule, nom, prenom, adresse, collaborateur*)
Inscriptions(matricule*, code*)
Modules(code, intitule, niveau, salle*)
Salles(numero, capacite, précédent *, suivante*)

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*)

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

Illustration évaluation requête 3

Etudiants

e →

matricule	nom	prénom	…
1753	Smith	Joe	
9832	Smith	Dan	
….			

Inscriptions

i →

Matricule*	Code*
1753	LI341
1753	LI345
9832	LI341

Modules

m →

code	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*)

4. a) Les intitulés des modules où 'Jack' est inscrit :

Evaluation requête 4-a

Etudiants

j →

matricule	nom	prénom	…
1753	Smith	Joe	
9832	Smith	Jack	
….			

Inscriptions

i →

Matricule*	Code*
1753	LI341
1753	LI345
9832	LI341

Modules

m →

code	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*)

4. b) Les étudiants inscrits dans un même module que 'Jack' ainsi que l'intitulé de ces modules :

Evaluation requête 4-b

Etudiants

	matricule	nom	prénom	…
e →	1753	Smith	Joe	
j →	9832	Smith	Jack	
	….			

Inscriptions

	Matricule*	Code*
i1 →	1753	LI341
	1753	LI345
i2 →	9832	LI341

Modules

	code	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*)

5. a) Les étudiants qui ne sont pas inscrits au module 'LI345'

Evaluation requête 5-a)

Etudiants

	matricule	nom	prénom	…
e →	1753	Smith	Joe	
	9832	Smith	Jack	
	….			

Inscriptions

	Matricule*	Code*
i →	1753	LI341
	1753	LI345
	9832	LI341

Différence

Etudiants(matricule, nom, prenom, adresse, collaborateur*)
Inscriptions(matricule*, code*)
Modules(code, intitule, niveau, salle*)
Salles(numero, capacite, précédent *, suivante*)

5. b) Les étudiants qui ne sont inscrits dans aucun module

Evaluation requête 5-b)

Etudiants

e →

matricule	nom	prénom	…
1753	Smith	Joe	
9832	Smith	Jack	
….			

Inscriptions

i →

Matricule*	Code*
1753	LI341
1753	LI345
9832	LI341

Modules

code	intitulé	niveau
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*)

6. Les étudiants inscrits dans tous les modules

Evaluation requête 6

Etudiants

e →

matricule	nom	prénom	…
1753	Smith	Joe	
9832	Smith	Jack	
….			

Inscriptions

i →

Matricule*	Code*
1753	LI341
1753	LI345
9832	LI341

Modules

m →

code	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*)

7. Les étudiants inscrits dans au moins deux modules

Evaluation requête 7

Etudiants

	matricule	nom	prénom	…
e →	1753	Smith	Joe	
	9832	Smith	Jack	
	….			

Inscriptions

	Matricule*	Code*
i1 →	1753	LI341
i2 →	1753	LI345
	9832	LI341

Modules

code	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*)

8. Les étudiants inscrits à au moins deux modules de niveau L3

Evaluation requête 8

Etudiants

	matricule	nom	prénom	…
e →	1753	Smith	Joe	
	9832	Smith	Jack	
	….			

Inscriptions

	Matricule*	Code*
i1 →	1753	LI341
	1753	M009
	9832	LI341
i2 →	1753	LI345

Modules

	code	intitulé	niveau
m1 →	LI341	BD	L3
m2 →	LI345	Web	L3
	LI399	Crypto	L3
	M009	BD avancées	M1

Jointure

Etudiants(matricule, nom, prenom, adresse, collaborateur*)
Inscriptions(matricule*, code*)
Modules(code, intitule, niveau, salle*)
Salles(numero, capacite, précédent *, suivante*)

9. a) Les étudiants inscrits à au moins deux modules de niveau L3 ou de niveau M1

Evaluation requête 9-a)

Etudiants

	matricule	nom	prénom	…
e →	1753	Smith	Joe	
	9832	Smith	Jack	
	….			

Inscriptions

	Matricule*	Code*
i1 →	1753	LI341
	1753	M009
	9832	LI341
i2 →	1753	LI345

Modules

	code	intitulé	niveau
m1 →	LI341	BD	L3
m2 →	LI345	Web	L3
	LI399	Crypto	L3
	M009	BD avancées	M1

Jointure

Etudiants(matricule, nom, prenom, adresse, collaborateur*)
Inscriptions(matricule*, code*)
Modules(code, intitule, niveau, salle*)
Salles(numero, capacite, précédent *, suivante*)

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

	matricule	nom	prénom	…
e →	1753	Smith	Joe	
	9832	Smith	Jack	
	….			

Inscriptions

	Matricule*	Code*
i1 →	1753	LI341
i →	1753	M009
	9832	LI341
i2 →	1753	LI345

Modules

	code	intitulé	niveau
m1 →	LI341	BD	L3
m2 →	LI345	Web	L3
	LI399	Crypto	L3
m →	M009	BD avancées	M1

Cas particuliers

- Requêtes sans réponse
  - { e | Etudiants(e) ∧ Modules (e) } : deux tables ne peuvent avoir exactement le même n-uplet (schéma différents)
  - {e |Etudiants(e) ∧ e.salaire>100} : salaire n'est pas attribut de Etudiants
- Requêtes avec réponse infinie → requête pas sûre
  - {e | ¬ Etudiants(e) } : chercher partout sauf dans la table Etudiants
  - {e | ∀ x Etudiants(x) ∧ … } : tous les n-uplets du monde doivent être dans Etudiant
  - {e | e.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 ∀, il faut toujours un ⇒ qui suit
  - Forme équivalente qu'on retrouve dans SQL:
    { t | Table(t) …. ∃ v ∈TableBis……¬ ∃ v ∈TableTer … }

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