Module 1 Algèbre Relationnelle

IGE 487 Modélisation de bases de données

Chargé de cours : Tudor Antohi

Mot introductif

Moi : passionné de base de données, science de données et Internet des objets

Vous : en train d'étudier votre cycle universitaire I et II

Mission : dans cette période de révolution déclenchée par l'intelligence artificielle et l'internet des objets, vous transmettre de connaissances qui vous aide a comprendre le rôle de bases de données, comment les modéliser et les utiliser dans vos besoins futures.

Règles

S’il vous plaît m’interrompre quand :

→ Je parle trop vite.

→ Tu ne comprends pas de quoi je parle.

→ Vous avez une question liée à la base de données.

Sinon posez les questions dans Teams et je vais répondre.

Cours Prérequis

IFT 187 – Éléments de bases de données

Concepts et architecture des bases de données. Création, interrogation et mise à jour d'une base de données relationnelle à l'aide du **langage SQL**. Requêtes complexes. **Contraintes d'intégrité**. **Modélisation entité-relation**. Traduction d'un modèle entité-relation en un modèle relationnel. **Dépendances fonctionnelles, dépendances multivaluées, dépendances de jointure. Normalisation : 1FN à 5FN et BCNF.**

IGE 287 – Exploitation des BDs relationnelles

Exploitation en mode client-serveur d'une base de données relationnelle et d'une base de données objet. Développement d'un système d'information simple. **Traitement de transactions. Accès concurrent aux données et préservation de l'intégrité des données.** Développement d'une application Web simple avec servlet et JSP. Échange électronique de données avec XML. Utilisation du langage Java comme environnement de programmation

Objectif de cours

Connaître

- les principes fondamentaux aux bases de données.
- les techniques de modélisation de bases de données
- les concepts de recherche et mise a jour de données
- le cycle de vie de données sur une approche temporelle

On vous prépare pour l'industrie & l'académie , pour le pragmatisme et pour la pensée abstraite.

Rôles dans les bases de données

DBA infrastructure – créé l'infrastructure

Concepteur de bases de données – en charge avec le concept de schéma

Développeur de bases de données – code applicatif et interne

DBA applicatif – optimisation de fonctionnement BD

Architectes et Analystes – évaluer la nécessité d'une certaine solution BD

Et aussi utilisateur de bases de données ☺

Bref

- que vous le sachiez ou non, vous utilisez une base de données toutes les ~~jours~~ heures

Contenu de cours

Contenu :

1. Algèbre relationnelle.
2. SQL Avancé, Modèle relationnel simple et relationnel étendu – normalisations 1NF-6NF
3. Schémas et les objets d'une BD
4. Optimisation des requêtes.
5. Transactions.
6. Contrôle de la concurrence.
7. Recouvrement, journalisation.
8. Conception d'une base de données distribuée.
9. Entrepôts et forage de données.
10. Base de données temporelles.

Attention : on sera puriste mais pas trop , la mise est sur une théorie riche en exemples. On forme de spécialistes capables de comprendre, réfléchir et appliquer.

Évaluation

Évaluation	%
Examen Intra	20
4 Devoirs TP1..4	30
Projet avec 4 jalons	30
Examen final	20

Détail	Date Début	Date fin
TP1 Algèbre	30-08	13-09
Projet J1,J2,J3, J4	30-08	
Projet J1		20 -09
TP2 Normalisation et schémas BD	06-09	27-09
Projet J2		10-04
Examen intra	18-10	
TP3 Optimisation	04-10	08-11
TP4 Temporalité	08-11	29-11
Projet J3		08-11
Projet J4		29-11
Examen final	13-12	

Composition de 3 cours de rappel

Introduction

Théorie relationnelle

Modelé entité - relationnel

Opérateurs relationnels

SQL

La normalisation

Schémas et objets d’une base de données

Le chemin vers la sagesse

Données – faits , Ex.: âges des employés 23, 45. Noms des employés : Émilie, Monique, Brian

Information – ensembles de données , *données + contexte* ***Ex.:*** *Émilie est 32*

Connaissance – prise de conscience a partir de données , *informations + analyses Ex.: L'âge moyenne de la retraite est 65*

Sagesse – capacité de décision (correcte) , *connaissance et jugement . Ex.: Brian prends bientôt sa retraite, on doit commencer le training pour d'autres employés*

Pourquoi modéliser

Deux écoles de pensée :

1. **Moderne** - Stocker le tout dans des collections de données sans une structure trop organisée , mettre des étiquettes et laisser l'intelligence applicative (artificielle ?) se rendre compte de meilleurs réponses.

Ex.: exploration d'un lac de données, stockages analytiques *one size fits it all.*

2. **Classique** - Stocker le tout dans des modèles des stockage optimisés et adaptés aaux besoins d'affaires pour aider avec les meilleures réponses. Ex.: notre cours.

But de la modélisation

- Rendre l'information plus claire en réduisant la redondance fonctionnelle de données et eliminant le risque d'incomplétude.

- Vision carboneutre
  - Réduire l'espace utilisé
  - Optimiser les mises a jours de données
  - Optimiser la performance de requêtes

<u>Désavantage</u>

On brule quelques neurones de plus ☺

Modélisations de données

- Modèle ontologique
- Modèle contextuel de données
- Modèle conceptuel de données
- **Modèle logique de données**
- **Modèle relationnel normalisé**
- **Modèle physique**

Rappel

Pour organiser et chercher une information dans une base de données, il faut comprendre :

- Logique propositionnelle
- Logique premier ordre
- Logique deuxième ordre
- Algèbre relationnelle
- Calcule relationnel

Rappel : logique d'ordre zéro

Déclarations simples , propositions élémentaires, combinaisons logiques

Expression	Observation
⌝A	Négation de A, si A est Vrai, le résultat est Faux
A ∧ B	Conjonction : Le résultat est vrai seulement si A et B sont vrais
A ∨ B	Disjonction : Le résultat est faux seulement si A et B sont faux
A → B	Implication : Si A alors B, attention, Faux peut impliquer Vrai mais Vrai ne peut pas impliquer Faux.
A ↔ B	Équivalence : les deux sont vraies ou faux
F ⊢ A	Déduction : de l'ensemble de formules F on déduit A
⊢ A	Théorème A

Lois de Morgan

$$\neg(p \wedge q) \vdash (\neg p \vee \neg q)$$

$$\neg(p \vee q) \vdash (\neg p \wedge \neg q)$$

Lois de l'équivalence

$$(p \leftrightarrow q) \vdash ((p \rightarrow q) \wedge (q \rightarrow p))$$

$$(p \leftrightarrow q) \vdash ((p \wedge q) \vee (\neg p \wedge \neg q))$$

$$(p \leftrightarrow q) \vdash ((p \vee \neg q) \wedge (\neg p \vee q))$$

Rappel : logique premier ordre

- Introduit les constants, variables
- Utilise des prédicats P , déclarations de propriétés et relations entre les objets, les phrases sont plus riches

Jean est marié avec Anne , et il possède une voiture rouge Ford 2019.

- Introduit les quantificateurs ($\exists$ existentiel, $\forall$ universel)

Déclaration	Vrai
$\forall x$ P(x)	P(x) vrai pour tout x
$\exists x$ P(x)	Il existe au moins un x qui fait que P(x) est vrai

Définitions – Ensembles des entités

Ensembles : collections de données qui n'ont pas un ordre défini, non-changeables et ne permettent pas de duplications (*sets* en anglais) {}

{"Québec", "Montréal"}

Sacs : collections de données qui n'ont pas un ordre défini, non-changeables et permettent de duplications (comme en SQL , pas trop puriste) (*bags* en anglais) {}

{"Québec", "Montréal" ,"Québec"}

Définitions – Théorie de types

Domaine : un ensemble de valeurs distinctes, noté avec **D** *Ex.: Integer N, Char C*

Contrainte : une expression logique entre les valeurs des identifiants qui doit être vraie ou faux

Ex.: X 2..10, X $\leq$ 99999999

Types : ensemble de valeurs défini par un domaine et une contrainte. Un **sous-type** (appelé aussi **type dérivé**) est un sous-ensemble d'un type, déterminé par une **contrainte** explicite. Noté aussi avec **D**

Ex.: $\{ x \in N \mid x \leq 99999999 \}$, $\{ x \in C^* \mid longueur\ (x) \leq 40 \}$

Attribut : couple formé d'un identifiant **a** et d'un type **D**, noté **(a:D).**

Ex.: *Identifiant :* ***a = Matricule,*** *Type :* ***D =*** $\{ x \in C^* \mid longueur\ (x) = 9 \}$

Attribut : (Matricule : $\{ x \in C^* \mid longueur\ (x) = 9 \}$*)*

Contrainte

Une **contrainte** est une expression logique applicable aux composants relationnels d'une base de données; l'expression doit être maintenue vraie tout au long de l'existence de la base de données

- **Contrainte de domaine** - restreindre l'ensemble des valeurs d'un domaine (pour en faire un type, par exemple)
- **Contrainte d'attribut** - limiter la valeur des attributs d'un tuple entre eux
- **Contrainte de relation** - limiter globalement les valeurs de la relation
- **Contrainte d'intégrité référentielle** - limiter globalement les valeurs d'attributs d'une relation relativement à une clé candidate d'une autre;

Définitions – Nulles

Un attribut prend une valeur nulle lorsqu'une entité n'a pas de valeur pour lui.

*La valeur nulle peut indiquer « **sans objet** », c'est-à-dire que la valeur n'existe pas pour l'entité. Par exemple, on peut ne pas avoir de deuxième prénom.*

*Null peut également indiquer qu'une **valeur d'attribut est inconnue**.*

*Une **valeur inconnue** peut être **manquante** (la valeur existe, mais nous n'avons pas cette information) ou **inconnue** (nous ne savons pas si la valeur existe réellement ou non).*

En général le message est :

La valeur nulle = la valeur n'existe pas dans la base de données et on ne sait pas pourquoi.

C'est une valeur commode qui s'applique indépendamment de typage des attributs.

On va voir dans le prochains cours que cette valeur crée des problèmes dans les opérateurs et on va discuter des solutions pratiques.

Définition – Tuple

Tuple : une composition d'attributs. Leurs types et leurs valeurs.

$t \triangleq (\{a1:D1, a2:D2, \ldots, a_n:Dn\} ; \{(a1,v1), (a2,v2), \ldots (a_n,v_n)\})$ avec $\forall i : 1 \leq i \leq deg(t) \Longrightarrow val(t, ai)\ \varepsilon\ def(t, ai)$

Obs.: La représentation est d'un couple de deux ensembles : types des identifiants et valeurs de identifiants.

Tuple : proposition modélisée par un ensemble d'attributs représenté par une ligne (un enregistrement).

Ex.:

Matricule : type Matricule	Nom : type Nom	Age : type Nombre naturel
151236239	Jean	25

Notations

$def(t) = \{a1:D1, a2:D2, \ldots, an:Dn\}$	entête de t
$def(t, ai) = Di$	type de ai
$val(t) = \{(a1,v1), (a2,v2), \ldots (an,vn)\}$	valeur de t
$val(t, ai) = vi$	valeur de ai
$t.ai = vi$	valeur de ai
$t(ai) = vi$	valeur de ai
$deg(t) = n$	degré de t
$id(t) = \{a1, a2, \ldots, an\}$	les identifiants d'attributs de t

Définition – Relations

Relation – Soit ai des identifiants distincts, D*j* des types et t*k* des tuples, une relation R est définie comme suit :

$R \triangleq (\{a1:D1, a2:D2, \ldots, an:Dn\} ; \{t1, t2, \ldots, tm\})$ avec $\forall i : 1 \leq i \leq card(R) \Rightarrow def(R)=def(ti)$

Notations

$def(R) = \{a1:D1, a2:D2, \ldots, an:Dn\}$	entête de R
$def(R, ai) = Di$	type de ai
$val(R) = \{t1, t2, \ldots, tm\}$	valeur de R
$deg(R) = n$	degré de R
$card(R) = m$	cardinalité de R
$id(R) = \{a1, a2, \ldots, an\}$	identifiants d'attributs de R

Relation : prédicat modélisé par un ensemble de tuples représentée par un tableau (une table).

Définition – Relations

Relvar – variable référant une (valeur de) relation.

Reltype – type de relation {a1:D1, a2:D2,..., a*n*:D*n*}

Relcon – contrainte de relation (**ex.: une clé unique pour les valeurs de Matricule**)

Ex. relvar:

matricule Matricule	nom Nom	addresse Ville
15113150	Paul	[illegible]
15112354	Éliane	Blanc-Sablon

Ex. reltype :

matricule Matricule	nom Nom	addresse Ville

Contrainte de relation

Clé d'une relation R - sous-ensemble d'attributs déterminant un tuple unique au sein de la relation.

`Soit X une clé de R : #R = #(R < {X})`

Clé référentielle (d'une relation R sur une relation S) - sous-ensembles d'attributs dont les valeurs sont restreintes par la clé d'une (autre) relation

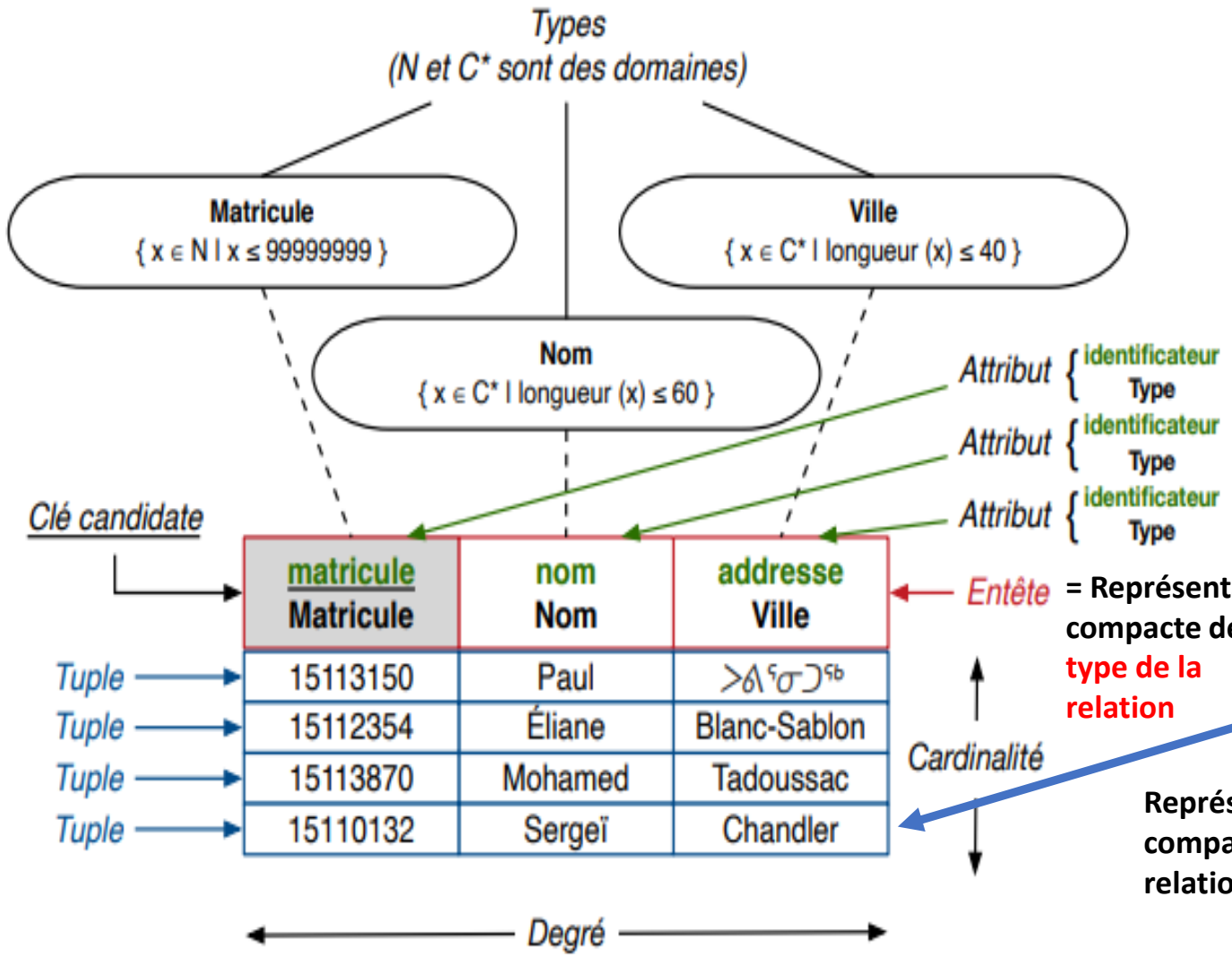
`Soit X une clé référentielle de R sur S : (R < {X}) ⊆ (S < {X}) et (X est une clé de S)`

Clé irréductible – clé dont on ne peut retirer aucun attribut sans qu'elle cesse d'être une clé.

Clé candidate : clé irréductible.

Surclé: tout ensemble d'attributs contenant une clé.

Ex.: Relation étudiant



Une relation comprenant quatre tuples

matricule : Matricule	nom : Nom	adresse : Ville	
matricule : Matricule matricule : 15113150	nom : Nom nom : Paul	adresse : Ville adresse : ⊁ϧ\ʕσ⊃ʢb	t1
matricule : Matricule matricule : 15112354	nom : Nom nom : Éliane	adresse : Ville adresse : Blanc-Sablon	t2
matricule : Matricule matricule : 15113870	nom : Nom nom : Mohamed	adresse : Ville adresse : Tadoussac	t3
matricule : Matricule matricule : 15110132	nom : Nom nom : Sergeï	adresse : Ville adresse : Chandler	t4

= Représentation compacte de type de la relation

Représentation compacte de la relation

Définition – Base de données

Soit **vi** des identifiants distincts, **D*j*** des types de relation et **r*k*** des (valeurs de) relations, **une base (de données) B** est définie comme suit :

$B \triangleq (\{v1:D1, v2:D2, \ldots, vn:Dn\} ; \{r1, r2, \ldots, rm\})$

avec $\forall i : 1 \leq i \leq card(B) \Rightarrow def(B, vi) = def(ri)$

Notations

$def(B) = \{v1:D1, v2:D2, \ldots, vn:Dn\}$	entête de B
$def(B, vi) = Di$	type de vi de B
$val(B) = \{r1, r2, \ldots, rm\}$	valeur de B
$deg(B) = n$	degré de B
$id(B) = \{v1, v2, \ldots, vn\}$	ensemble des identifiants de variables de relation de B

Schéma : Soit **ai** des identifiants distincts, **D*j*** des types de relation , un schéma est défini comme suit :
S ≜{a1:D1, a2:D2,…, an:Dn}
Un schéma : un ensemble de définitions de relvars et un ensemble de définitions de contraintes. *Le schéma est le type de la base de données.*

Base de données *= Un ensemble de relvars conformes à un schéma (donc aux définitions des relvars et des contraintes). Elle peut être définie à l'aide de plusieurs schémas; chaque identifiant défini par un schéma est alors qualifié (préfixé) par le nom du schéma.*

Ex.: Base de données

- Plusieurs relations , plusieurs tables , plusieurs ensembles de relations
- Expliquez en classe sur l’image de PostgreSQL

Les relations et la logique

relation	prédicat modélisé par un ensemble de tuples représentée par un tableau (une table).
tuple	proposition modélisée par un ensemble d'attributs représenté par une ligne (un enregistrement).
attribut	variable typée représentée par une cellule (un champ).
contrainte	expression logique.
relvar	variable référant une (valeur de) relation.
schéma	un ensemble de définitions de relvar et un ensemble de définitions de contraintes.
base de données	un ensemble de relvars conformes à un schéma (donc aux définitions des relvars et des contraintes).

Vers le modèle info-relationnel

- **Redéfinition pragmatique**

Une base de données est une collection organisée de données interdépendantes qui modélise certains aspects du monde réel

Un **système de gestion de base de données** (**SGBD**) est le logiciel qui gère une base de données. Il est conçu pour permettre la définition, création, interrogation, mise à jour et l'administration des bases de données.

Ex. SGBD : MySQL, PostgreSQL, SQL Server, Oracle

Concepts clés : Modèle de données, Schémas vs Données, DDL, DML

Les défis de SGBDs

- ✓Massives – térabytes, pétabytes
- ✓Persistance
- ✓Sécurité – matériel, logiciel, alimentation, utilisateurs
- ✓Multi-utilisateur – contrôle de la concurrence
- ✓Facilité d'utilisation – langages , indépendance de la couche physique
- ✓Efficience – milliers de requêtes par seconde
- ✓Fiabilité – Ex.: 99,99999

Modèle de données

Un **modèle de données** est un ensemble de concepts permettant de décrire les données d'une base de données.

Un **schéma** est une description d'une collection particulière de données, à l'aide d'un modèle de données donné

Les **relations** sont des **tables** dans un SGBD.

Dans la modélisation **E-R** on utilise souvent le mot **entité**

Types de modèles de données

A. **Relationnel – gagne presque toujours**
B. Clé/Valeur
C. Graph
D. Document
E. Famille de colonnes
F. Tableau / Matrice
G. Réseau hiérarchique
H. Multi-Valeur
I. Multi-modèles …

Modèles de données relationnel

Structure : Définition des relations de la base de données et de leur contenu.

Intégrité : Le contenu de la base de données respecte les contraintes.

Manipulation de données : Façon d'accéder au contenu d'une base de données et de le modifier.

Modèles relationnel : clés primaires

La clé primaire d'une relation (table) identifie de manière unique un seul tuple.

Certains SGBD créent automatiquement une clé primaire interne si une table ne l'a pas définie

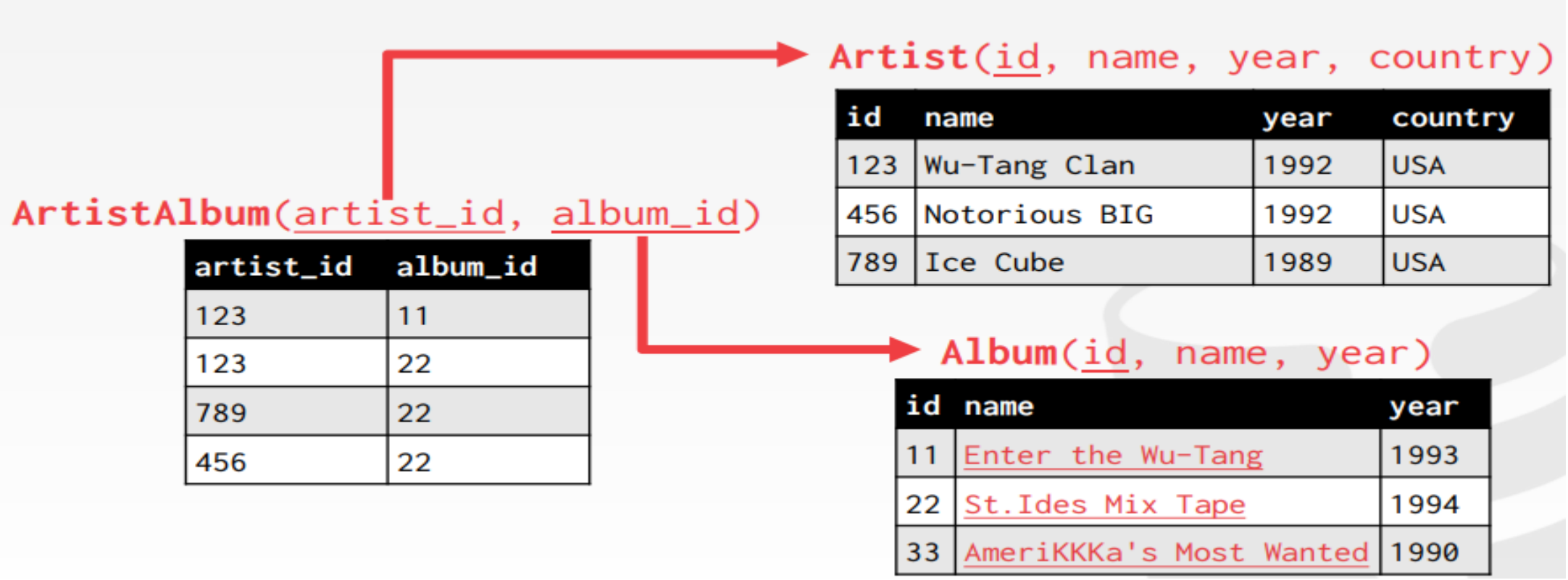
Génération automatique de clés primaires entières uniques :

→ SÉQUENCE → (SQL:2003)

→ AUTO_INCREMENT (MySQL)

Modèles relationnel : clés référentielles

Une clé référentielle spécifie qu'un attribut d'une relation (table) doit correspondre à un tuple dans une autre relation (table)



Langages de manipulation de données

Méthodes de stockage et interrogation d'informations à partir d'une base de données.

<u>Procédurales , basées sur l'algèbre relationnelle</u>

La requête spécifie la stratégie (de haut niveau) que le SGBD doit utiliser pour trouver le résultat souhaité.

<u>Non-procéduralles , calcule relationnel – comme SQL</u>

La requête spécifie uniquement les données souhaitées et non la manière de les trouver

Opérateurs relationnels

- **Opérateurs de base**, seront définit utilisant la théorie d'ensembles
- **Opérateurs usuels,** seront définit par des expressions booléennes simples a partir des opérateurs de base, exemples SQL
- **Opérateurs spécialisés,** seront définis par des expressions booléennes simples et avec des exemples SQL
- **Opérateurs de groupement,** seront définis par des expressions booléennes simples et avec des exemples SQL

Opérateurs relationnels de base

6 opérateurs de base, a partir de quelles toutes les autres opérations sur les relations sont composables.

1. Renommage, *ρ (A : B) :* la relation comprenant les tuples formés à partir d'un tuple de **R** dont l'attribut de nom **A** est remplacé par un attribut de nom **B** de même valeur.

R ρ a:b ≡ R RENAME {a AS b} ≡ ANTE a ∈ id(R) ⋀ b ∉ id(R) :
SOIT E := def(R,a) Z := def(R) - {a:E} ∪ {b:E}
: (Z ; {(Z ; val(t) - {(a,val(t,a))} ∪ {(b,val(t,a))}) | t ∈ val(R)})

2. Restriction, R σ c : la relation comprenant les tuples de **R** satisfaisant la condition **c**

R σ c ≡ R WHERE c ≡ ANTE id(c) ⊆ id(R)
: (def(R); {t | t ∈ val(R) ⋀ c(t)})

3. Projection, R π x : la relation comprenant les tuples formés à partir d'un tuple de **R** dont seuls les attributs dont le nom est parmi **x** ont été conservés

R π x ≡ R {w0, ..., wn} ≡ ANTE w ⊆ {a | a:D ∈ def(R)} : SOIT Z := {a:D | a:D ∈ def(R) ⋀ a ∈ w}
: (Z ; {(Z ; {(a,v) | (a,v) ∈ val(t) ⋀ a ∈ w}) | t ∈ val(R)})

Opérateurs relationnels de base

6 opérateurs de base, a partir de quelles toutes les autres opérations sur les relations sont composables.

4. Jointure, R ⋈ S : la relation comprenant tous les tuples formés à partir d'un tuple de **R** et d'un tuple de **S** dont les attributs de même nom sont de même valeur

R ⋈ S ≡ R JOIN S ≡ SOIT Z := def(R) ∪ def(S) x := id(R) ∩ id(S) : ANTE ∀ a ∈ x : a:D ∈ def(R) ⋀ a:D ∈ def(S)

: (Z ; {(Z ; val(t1) ∪ val(t2)) | t1 ∈ val(R) ⋀ t2 ∈ val(S) ⋀ (∀ a ∈ x : (a,v) ∈ val(t1) ⋀ (a,v) ∈ val(t2))}).

5. Union, R ∪ S : la relation comprenant tous les tuples de **R** et tous les tuples de **S** sans duplications, et rien d'autre

R ∪ S ≡ R UNION S ≡ ANTE def(R) = def(S)

: (def(R); val(R) ∪ val(S))

6. Différence, R – S : la relation comprenant tous les tuples de **R** qui ne sont pas dans **S**, et rien d'autre.

R – S ≡ R MINUS S ≡ ANTE def(R) = def(S)

: (def(R); val(R) – val(S))

Exemple : Opérateurs relationnels de base

Restriction
R σ cond

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3

Projection
R π {A, C}

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3

Jointure (naturelle)
R ⋈ S

A	B
a1	b1
a2	b1
a3	b3
a4	b4

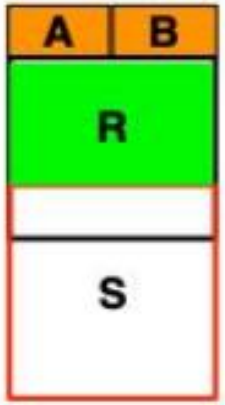
⋈

B	C
b1	c1
b2	c2
b3	c3
b3	c4

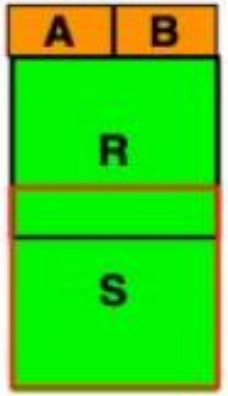
=

A	B	C
a1	b1	c1
a2	b1	c1
a3	b3	c3
a3	b3	c4

Différence
R – S



Union
R ∪ S



Renommage
R ρ A:C

A	B
a1	b1
a2	b2
a3	b3

ρ A:C =

C	B
a1	b1
a2	b2
a3	b3

Exemple : Opérateurs relationnels de base

Université (**Nom Université**, Provence, Nombre Étudiants)

Étudiant (**Id Et**, Nom Et, Moyenne Examen ME, Moyenne Collège MC)

Demande (**Id Et, Nom Université**, Programme, Acceptation)

Noms er moyenne des examens des étudiants avec une moyenne collège MC plus de 25 qui ont appliqué a IGE487 et ont été rejetés.

$\pi_{\text{Id Et, ME}}$ ($\sigma_{\text{Étudiant.Id Et = Demande.Id Et} \wedge \text{ Programme = IGE487} \wedge \text{ Acceptation = 'R'}}$ (Étudiant ⋈ Demande))

Question 1

Énonce A :

C’est utile de composer deux opérateurs de sélection.

$\sigma_{\text{cours = IGE487}} (\sigma_{\text{étudiant = 2}} \text{Université})$

Énoncé B :

C’est utile de composer deux opérateurs de projection.

$\pi_{\text{cours}} (\pi_{\text{étudiant, session}} \text{Université})$

C’est quoi la vérité ?

Question 1 – Réponse

C’est inutile de composer ces 2 opérateurs, c’est plus de travail iunutile pour le moteur BD a réfléchir et réécrire la requête.

Ex.: Quand tu cherche dans une bibliothèque tu ne sors pas tous les livres d’un certain auteur pour une 2-eme recherche pour le titre.

$\sigma_{\text{cours} = \text{IGE487}}(\sigma_{\text{étudiant} = 2}\text{Université}) \equiv \sigma_{(\text{cours} = \text{IGE487} \wedge \text{étudiant} = 2)}\text{Université}$

$\pi_{\text{cours}}(\pi_{\text{étudiant, session}}\text{Université}) \equiv$ ne marche pas. La première projection sélectionne des attributs qui n’existe pas.

Question 2

```
College(cName,state,enrollment)
Student(sID,sName,GPA,sizeHS)
Apply(sID,cName,major,decision)
```

HS = lycée
sID = ID étudiant
cName = nom collège
sName = nom étudiant
sizeHS = nombre des étudiants lycée
Enrollment = nombre des étudiants collège
Major = spécialité

Laquelle des expressions suivantes NE donne PAS les noms et les GPAs des étudiants avec une sizeHS supérieur à 1000 qui ont postulé à CS et ont été rejetés

- $\pi_{sName,GPA}\left(\sigma_{Student.sID=Apply.sID}\left(\sigma_{HS>1000}\left(Student\right)\times\sigma_{major='CS'\wedge dec='R'}\left(Apply\right)\right)\right)$
- $\pi_{sName,GPA}\left(\sigma_{Student.sID=Apply.sID\wedge HS>1000\wedge major='CS'\wedge dec='R'}\left(Student\times\pi_{sID,major,dec}Apply\right)\right)$
- $\sigma_{Student.sID=Apply.sID}\left(\pi_{sName,GPA}\left(\sigma_{HS>1000}Student\times\sigma_{major='CS'\wedge dec='R'}Apply\right)\right)$

Question 2 – Réponse

College(**cName**,**state**,**enrollment**)
Student(**sID**,**sName**,**GPA**,**sizeHS**)
Apply(**sID**,**cName**,**major**,**decision**)

On observe que la 3-eme opération ne peut pas appliquer la restriction parce que la projection sélectionne d'autres colonnes.

- $\pi_{sName,GPA}\left(\sigma_{Student.sID=Apply.sID}\left(\sigma_{HS>1000}\left(Student\right)\times\sigma_{major='CS'\wedge dec='R'}\left(Apply\right)\right)\right)$
- $\pi_{sName,GPA}\left(\sigma_{Student.sID=Apply.sID\wedge HS>1000\wedge major='CS'\wedge dec='R'}\left(Student\times\pi_{sID,major,dec}Apply\right)\right)$
- $\sigma_{Student.sID=Apply.sID}\left(\pi_{sName,GPA}\left(\sigma_{HS>1000}Student\times\sigma_{major='CS'\wedge dec='R'}Apply\right)\right)$

Question 3

College(cName,state,enrollment)
Student(sID,sName,GPA,sizeHS)
Apply(sID,cName,major,decision)

HS = lycée
sID = ID étudiant
cName = nom collège
sName = nom étudiant
sizeHS = nombre des étudiants lycée
Enrollment = nombre des étudiants collège
Major = spécialité
CS = informatique

Comment decrire le resultat suivant .

$$\pi_{sName,cName}\left(\sigma_{HS>enr}\left(\sigma_{state='CA'}College \bowtie Student \bowtie \sigma_{major='CS'}Apply\right)\right)$$

1. toutes les paires de noms de collèges et étudiants, où l'étudiant postule pour se spécialiser en informatique au collège, le collège est en Californie et le collège est plus petit que certains lycées
2. étudiants jumelés avec tous les collèges californiens auxquels l'étudiant a postulé pour se spécialiser en informatique, où au moins un de ces collèges est plus petit que le lycée de l'étudiant
3. étudiants jumelés avec tous les collèges plus petits que le lycée de l'étudiant auquel l'étudiant a postulé pour major en CS où au moins un de ces collèges se trouve en Californie
4. étudiants jumelés avec tous les collèges californiens plus petits que le lycée de l'étudiant auquel l'étudiant a postulé pour se spécialiser en informatique

Question 3 – Réponse

College(cName,state,enrollment)
Student(sID,sName,GPA,sizeHS)
Apply(sID,cName,major,decision)

Comment decrire le resultat suivant .

$$\pi_{sName,cName}\left(\sigma_{HS>enr}\left(\sigma_{state='CA'}College \bowtie Student \bowtie \sigma_{major='CS'}Apply\right)\right)$$

Réponse : première jointure sélectionne tous les collèges et les étudiants de l'état de Californie, c'est un produit cartésien. Il n'existe pas des attributs communs. La deuxième jointure , sélectionne tous les étudiants qui ont appliqués pour un major en CS. Ils peuvent appliquer aux plusieurs collèges.

Cette fois la jointure se fait sur **sID et cName**.

La troisième restriction demande que le nombre des étudiant de lycée soit plus grand que le nombre des étudiants de collège.

IL faut faire attention que la jointure sur cName n'exclue pas le fait que certains collèges avec le même **cName** qui ne sont pas en **Californie** sont quand même sur la liste.

Question 3 – Réponse analyse (1)

College(cName,state,enrollment)
Student(sID,sName,GPA,sizeHS)
Apply(sID,cName,major,decision)

Comment decrire le resultat suivant .

$$\pi_{sName,cName}\left(\sigma_{HS>enr}\left(\sigma_{state='CA'}College \bowtie Student \bowtie \sigma_{major='CS'}Apply\right)\right)$$

Réponse : première jointure sélectionne tous les collèges et les étudiants de l'état de Californie, c'est un produit cartésien. Il n'existe pas des attributs communs. La deuxième jointure , sélectionne tous les étudiants qui ont appliqués pour un major en CS. Cette fois la jointure se fait sur sID et cName.

La troisième restriction demande que le nombre des étudiant de lycée soit plus grand que le nombre des étudiants de collège. **Un ne se qualifie pas** parce qu'il s'agit pas de seulement certains collèges. La requête et pour tous les lycées, il n'existe pas de restrictions.

1. toutes les paires de noms de collèges et étudiants, où l'étudiant postule pour se spécialiser en informatique au collège, le collège est en Californie et le collège est plus petit que certains lycées

Question 3 – Réponse analyse (2)

College(cName,state,enrollment)
Student(sID,sName,GPA,sizeHS)
Apply(sID,cName,major,decision)

Comment decrire le resultat suivant .

$$\pi_{sName,cName}\left(\sigma_{HS>enr}\left(\sigma_{state='CA'}College \bowtie Student \bowtie \sigma_{major='CS'}Apply\right)\right)$$

Réponse : première jointure sélectionne tous les collèges et les étudiants de l'état de Californie, c'est un produit cartésien. Il n'existe pas des attributs communs. La deuxième jointure , sélectionne tous les étudiants qui ont appliqués pour un major en CS. Cette fois la jointure se fait sur sID et cName.

La troisième restriction demande que le nombre des étudiant de lycée soit plus grand que le nombre des étudiants de collège. **Deux ne se qualifie pas** parce qu'il s'agit pas de au moins un collège. La requête est pour tous les lycées, il n'existe pas de restrictions.

2. étudiants jumelés avec tous les collèges californiens auxquels l'étudiant a postulé pour se spécialiser en informatique, où au moins un de ces collèges est plus petit que le lycée de l'étudiant

Question 3 – Réponse analyse (3)

College(cName,state,enrollment)
Student(sID,sName,GPA,sizeHS)
Apply(sID,cName,major,decision)

Comment decrire le resultat suivant .

$$\pi_{sName,cName}\left(\sigma_{HS>enr}\left(\sigma_{state='CA'}College \bowtie Student \bowtie \sigma_{major='CS'}Apply\right)\right)$$

Réponse : première jointure sélectionne tous les collèges et les étudiants de l'état de Californie, c'est un produit cartésien. Il n'existe pas des attributs communs. La deuxième jointure , sélectionne tous les étudiants qui ont appliqués pour un major en CS. Cette fois la jointure se fait sur sID et cName.

La troisième restriction demande que le nombre des étudiant de lycée soit plus grand que le nombre des étudiants de collège. Trois, bon! , la liste ne contient pas seulement les collèges Californiennes mais au mois un est sur la liste.

3. étudiants jumelés avec tous les collèges plus petits que le lycée de l'étudiant auquel l'étudiant a postulé pour major en CS où au moins un de ces collèges se trouve en Californie

Question 3 – Réponse analyse (4)

College(cName,state,enrollment)
Student(sID,sName,GPA,sizeHS)
Apply(sID,cName,major,decision)

Comment decrire le resultat suivant .

$$\pi_{sName,cName}\left(\sigma_{HS>enr}\left(\sigma_{state='CA'}College \bowtie Student \bowtie \sigma_{major='CS'}Apply\right)\right)$$

Réponse : première jointure sélectionne tous les collèges et les étudiants de l'état de Californie, c'est un produit cartésien. Il n'existe pas des attributs communs. La deuxième jointure , sélectionne tous les étudiants qui ont appliqués pour un major en CS. Cette fois la jointure se fait sur sID et cName.

La troisième restriction demande que le nombre des étudiant de lycée soit plus grand que le nombre des étudiants de collège. **Quatre n'est pas bon.** Comme mentionné, la jointure sur cName ne garanti pas que le résultat est pour **tous** les collèges de Californie. On a des étudiants qui n'ont pas appliqué en Californie.

4. étudiants jumelés avec tous les collèges californiens plus petits que le lycée de l'étudiant auquel l'étudiant a postulé pour se spécialiser en informatique

Question 4

Student(sID, sName, GPA, sizeHS)

La relation **Student** a 20 tuples. C'est quoi le min et le max de tuples qui résultent de cette expression.

$$\rho_{s1(i1,n1,g,h)} Student \bowtie \rho_{s2(i2,n2,g,h)} Student$$

1. Minimum = 0, maximum = 400
2. Minimum = 20, maximum = 20
3. Minimum = 20, maximum = 400
4. Minimum = 40, Maximum = 40

Question 4 – Réponse

Student(sID, sName, GPA, sizeHS)

La relation **Student** a 20 tuples. C'est quoi le min et le max de tuples qui résultent de cette expression.

$$\rho_{s1(i1,n1,g,h)} Student \bowtie \rho_{s2(i2,n2,g,h)} Student$$

Réponse: Minimum = 20, maximum = 400

Explication : le renommage force la jointure cartésienne seulement sur les valeurs des attributs **g** et **h**.
Si toutes les valeurs des attributs g et h sont différentes, on va avoir le résultat 20 (évidemment 1 fois, les même tuples seront sélectionnées pour 20 occurrences).
Si tous les valeurs des attributs g et h sont égales, on va avoir 20 x 20 = 400. Tous les tuples seront sélectionnés.

Question 5

College(<u>cName</u>,state,enrollment)	5 tuples
Student(<u>sID</u>,sName,GPA,sizeHS)	20 tuples
Apply(<u>sID</u>,<u>cName</u>,<u>major</u>,decision)	50 tuples

Si on assume que les noms de collèges en Apply apparaissent en Collège, c'est quoi le nombre minimum et maximum de tuples qui résultent de cette expression.

$$\pi_{cName} College \cup \rho_{cName} (\pi_{sName} Student) \cup \pi_{cName} Apply$$

1. Minimum 5, Maximum 25
2. Minimum 5 Maximum75
3. Minimum 25 Maximum 45
4. Minimum 75 Maximum 75

Question 5 – Réponse

College(<u>cName</u>,state,enrollment) 5 tuples
Student(<u>sID</u>,sName,GPA,sizeHS) 20 tuples
Apply(<u>sID</u>,<u>cName</u>,<u>major</u>,decision) 50 tuples

Si on assume que les noms de collèges en Apply apparaissent en Collège, c'est quoi le nombre minimum et maximum de tuples qui résultent de cette expression.

$$\pi_{cName} College \cup \rho_{cName} (\pi_{sName} Student) \cup \pi_{cName} Apply$$

Minimum 5, Maximum 25

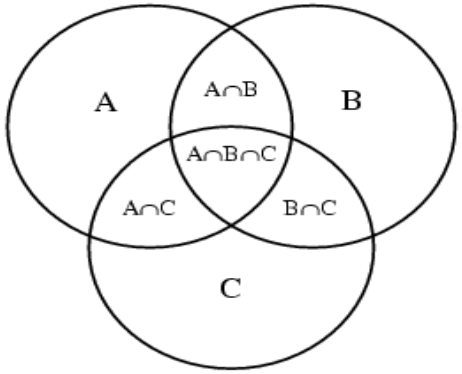
Réponse:
Le UNION de relations **Collège** et **Apply** , donnes 5 valeurs distinctes, car on a seulement 5 collèges distinctes dans **Collège** (regardez la clé **cName**)

Le UNION avec **Student,** donnes maximum 25 valeurs. Il est possible par pure chance d'avoir les noms de collège dans tous les noms d'étudiants et on va avoir seulement 5. Mais s'il n'y aucune coïncidence on fini par avoir 25. Ex.: Bishop peut être un nom étudiant ou un nom de collège.

Opérateurs usuels

1. **Intersection, ∩, INTERSECT**

R1 ∩ R2 = R1 – (R1 - R2)



2. Produit, R × S, TIMES

Produit cartesien

```
R ( a, b, c )        S ( b, c, d )
  ( 1, 2, 3 )          ( 2, 7, 9 )
  ( 2, 4, 6 )          ( 5, 3, 4 )
  ( 3, 6, 9 )          ( 2, 3, 6 )
```

```
( R.a, R.b, R.c, S.b, S.c, S.d )
( 1,   2,   3,   2,   7,   9   )
( 1,   2,   3,   5,   3,   4   )
( 1,   2,   3,   2,   3,   6   )
( 2,   4,   6,   2,   7,   9   )
( 2,   4,   6,   5,   3,   4   )
( 2,   4,   6,   2,   3,   6   )
( 3,   6,   9,   2,   7,   9   )
( 3,   6,   9,   5,   3,   4   )
( 3,   6,   9,   2,   3,   6   )
```

3. Semi-jointure, ⋉, MATCHING

R ⋉ S = (R ⋈ S) π R

Semi-jointure (matching)

R ⋉ S = (R ⋈ S) π R

A	B
a1	b1
a2	b1
a3	b3
a4	b4

⋉

B	C
b1	c1
b2	c2
b3	c3
b3	c4

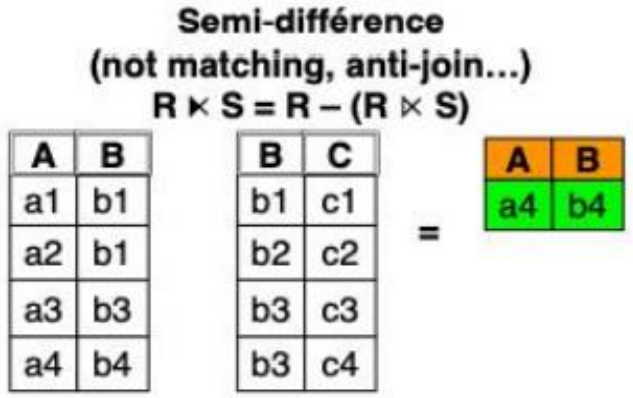
=

A	B
a1	b1
a2	b1
a3	b3

Opérateurs usuels

4. Semi-différence, ▷, NOT MATCHING

R ▷ S = R – (R ⋉ S)



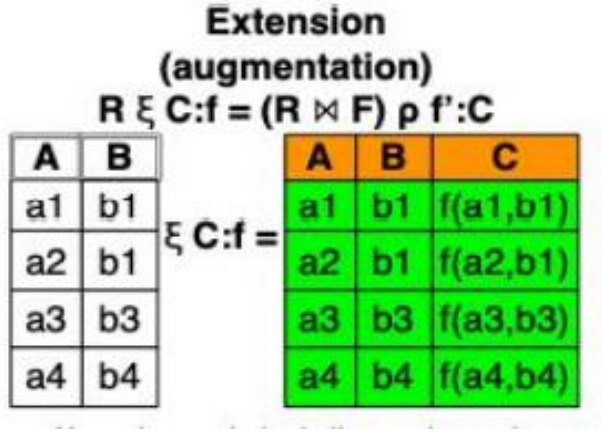
5. Extension, ξ, EXTEND

Extension avec un attribut qui contient une formule :

R ξ C:f = R × C les tuples C calculés par la formule f

EXTEND S ADD (3 * STATUS AS TRIPLE)

S#	SNAME	STATUS	CITY	TRIPLE
S1	Smith	20	London	60
S2	Jones	10	Paris	30
S3	Blake	30	Paris	90
S4	Clark	20	London	60
S5	Adams	30	Athens	90



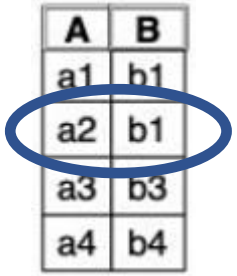
Opérateurs usuels

6. Image, ∘, IMAGE_IN

R ∘ t ≡ R(t) ≡ IMAGE t IN R ≡ R ⋈ RELATION[t π (id(t) ∩ id(R))]

val(tuple t) = {(A,a2), (B,b1)}

On le trouve dans la relation R



A	B
a1	b1
a2	b1
a3	b3
a4	b4

7. Image du tuple courant, !!

!! R ≡ R ∘ TUPLE[*]

Opérateurs spécialisés

1. **Union exclusive, ⊍ (UNION_X), XUNION**

R ⊍ S = (R – S) ∪ (S – R) = (R ∪ S) – (R ∩ S)

2. Union disjointe, ⊎ (UNION_D), D_UNION

a) Après avoir renommés les attributs communes de R ∩ S en S.

b) Si W = R ∩ S et $\mathrm{id}(\mathrm{R} \cap \mathrm{S}) = \{\mathrm{w1}, \mathrm{w2}, \ldots, \mathrm{w}n\}$

c) Renommage en S : *S ρ w1:w11, S ρ w2:w12, …, S ρ wn:wnn*,

d) Finalement R ⊎ S ≡ R x S

Opérateurs spécialisés

3. Différence inclusive, ⊖ (MINUS_I), I_MINUS

On garde en R les valeurs qui n'existe pas en S et tous les attributs de R restent en R.

a. Après avoir éliminer les attributs de **R ∩ S** en **R** on obtient **R'** avec **def(R') = def (S)**

b. Finalement **R ⊖ S ≡ (R' – S) ⋈ R**

Opérateurs spécialisés

4. Composition, •, COMPOSE

Les attributs communes de R et S sont supprimés dans la jointure naturelle effectuée.

Pensez comment réaliser 1,2,3,4 a partir des opérateurs de base.

Opérateurs spécialisés

5. Fermeture positive, FP

La fermeture peut être généralisée pour une relation binaire arbitraire **R ⊆ S×S**, et une propriété arbitraire P de la manière suivante : la fermeture P de R est la moindre relation **Q ⊆ S×S** qui contient R (c'est-à-dire R ⊆ Q) et pour laquelle **P(Q)** est vraie.

Ex.: *Les nombres réels sont fermés dans la propriété soustraction, mais les nombres naturels ne le sont pas : 3 et 8 sont tous deux des nombres naturels, mais le résultat de 3 – 8 n'est pas un entier naturel. Bref le sous-set de paires de nombre naturels **(a,b)** avec la contrainte **a** plus grand que **b** est une fermeture positive et il faut le trouver avec cet opérateur.*

Opérateurs spécialisés

6. Fermeture transitive, ⋆, TCLOSE

Soit R une relation avec des attributs **X** et **Y**, du même type **T**. Alors la fermeture **transitive** de **R**, TCLOSE R, est une relation **R +** dont l'entête est le même que celui de R et le corps un **sur-ensemble** de celui de R, définie comme suit : Le tuple **{ X x, Y y }** apparaît dans R + si et seulement s'il apparaît dans R ou s'il existe une séquence de valeurs **z1, z2, ..., zn**, de type **T**, telles que les tuples ***{ X x, Y z1 }, { X z1, Y z2 }, ..., { X zn, Y y }***.

Ex.: *La propriété "mes ancêtres » n'est pas fermée sur la propriété **ancêtre roumain**. Il est possible que j'ai un ancêtre qui est d'origine hongroise dans tous mes ancêtres depuis Adam. Trouvez donc le dernier ancêtre roumain et fermer la relation avec cette propriété. La propriété **ancêtre** est transitive et j'ai un exemple de fermeture positive d'une transition quand je trouve mes ancêtres roumains et je m'arrête.*

Opérateurs spécialisés

6 bis. Fermeture transitive, ⋆, TCLOSE

Relation R simple

Nom	Parent
Theodore	Rosemarie
Theodore	Emile
Emile	Basile
Emile	Catherine
Rosemarie	Nicholas
Rosemarie	Helene

Relation après TCLOSE qui trouve les ancêtres roumaines et s'arrête aux ancêtres émigrés de Grèce.

Nom	Parent
Theodore	Rosemarie
Theodore	Emile
Theodore	Basile
Theodore	Catherine
Theodore	Nicholas
Theodore	Helene
….	…

Opérateurs de groupement

WRAP Groupement par tuple

w tuple

a attribut utilisé pour créer le tuple

R relation

R WRAP w AS a ≡ ANTE w ⊆ id(R) ∧ a ∉ id(R)

: (R ξ {a := TUPLE [w1:w1, ..., wn:wn]) π \w

r	A	B	C
	b1	40	a1
	b1	40	a2
	b2	50	a3
	b3	60	a4
	b3	60	a2
	b3	60	a5
	b4	60	a6

temp1	A	B	C
	b1	40	{a1, a2}
	b2	50	{a3}
	b3	60	{a2, a4, a5}
	b4	60	{a6}

Opérateurs de groupement

UNWRAP Groupement par tuple

w tuple produit

a attribut a recréer

R relation

R UNWRAP a ≡ ANTE a ∈ id(R) ∧ « a est de type TUPLE » ∧ id(a) ∩ id(R) = ∅

: SOIT id(def(a)) = {w1, ..., wn} : R ξ {w1 := a π {w1}, ..., wn := a π {wn}} π \{a}

temp1	A	B	C
	b1	40	{a1, a2}
	b2	50	{a3}
	b3	60	{a2, a4, a5}
	b4	60	{a6}

r	A	B	C
	b1	40	a1
	b1	40	a2
	b2	50	a3
	b3	60	a4
	b3	60	a2
	b3	60	a5
	b4	60	a6

Opérateurs de groupement

Groupement par relation GROUP

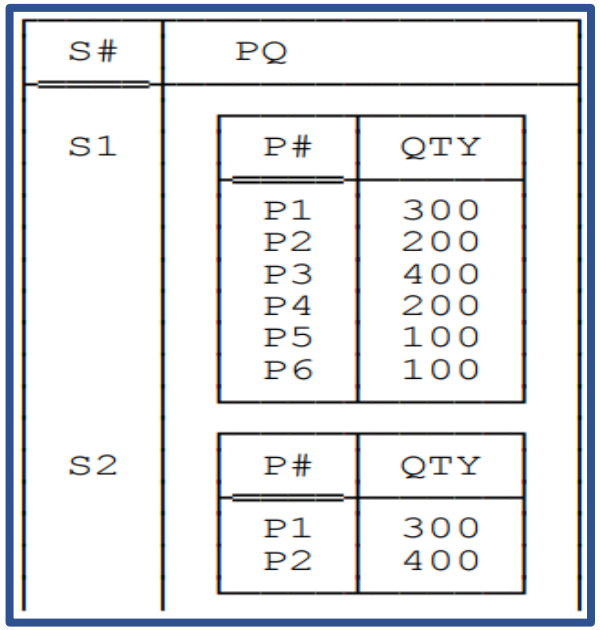
Au lieu de réutiliser un attribut comme dans WRAP, on génère un nouvel attribut **a qui contient le tuple w.**

R GROUP w AS a ≡ ANTE w ⊆ id(R) ∧ a ∉ id(R)

: R π \w ξ {a := !!R}

Ex.: sur une relation **def(SP) = {S:D1, P:D2, QTY:D3}**

SPQ = SP GROUP ({ P#, QTY } AS PQ)



S#	PQ
S1	P# / QTY P1 300 P2 200 P3 400 P4 200 P5 100 P6 100
S2	P# / QTY P1 300 P2 400

Opérateurs de groupement

Groupement par relation

UNGROUP

On utile un attribut **a** qui contient un tuple **w**, et on génère les attributs w1, w2, …, wn du **tuple w.**

SP = SPQ UNGROUP ({ P#, QTY } AS PQ)

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400

Opérateurs de groupement

Réfléchir :

Dans quel cas d’affaire, les opérateurs de groupement sont utiles ?

Opérateurs d'agrégation

Les plus connus opérateurs utilisés sont MIN, MAX, AVG, SUM et COUNT. Les attributs de regroupement forcent une fragmentation de la relation, la fonction est calculée dans chaque groupe indépendant. La sortie est constituée des attributs de regroupement et du résultat des fonctions sur groupe. Si aucun attribut de regroupement n'est donné, les fonctions s'appliquent à l'ensemble des attributs de la table.

$$\textbf{Temp} = \tilde{}_{\textbf{Sum(B),Max(C)}}\ \textbf{R}$$

r	A	B	C
	1	10	1
	1	2	5
	2	3	3
	3	6	10
	3	5	7

Temp	A	Sum_B	Max_C
	1	12	5
	2	3	3
	3	11	10

Équivalences SQL en PostgreSQL (1)

Notion	Exemples SQL
Attribute	Colone dans une table avec le type associé
Tuple	Une rangée dans une table
Relation	Vue ou Table
Contrainte, Clé unique, Clé candidate, Clé référenetielle	Correspondance similaire
Renommage	SELECT field AS field1 FROM … ALTER TABLE table_name RENAME COLUMN column_name TO new_column_name;
Restriction	SELECT * FROM table_name WHERE column_name = ‘Sherbrooke’
Projection	SELECT column1, column2 FROM table_name
Jointure	SELECT * FROM table_name1 INNER JOIN table_name2
Union	SELECT * FROM table_name1 UNION SELECT * FROM table_name2
Différence	SELECT * FROM table_name1 MINUS SELECT * FROM table_name2

Équivalences SQL en PostgreSQL (2)

Notion	Exemples SQL
Intersection	SELECT select_list FROM A INTERSECT SELECT select_list FROM B;
Produit	SELECT * FROM table1 CROSS JOIN table2;
Semi-jointure	SELECT DISTINCT s.id FROM students s LEFT JOIN grades g ON g.student_id = s.id WHERE g.student_id IS NOT NULL
Semi-différence (anti-join)	SELECT l.id, l.value FROM t_left l LEFT JOIN t_right r ON r.value = l.value WHERE r.value IS NULL
Extension	SELECT t.*, (t.n1+t.n2) as ext FROM table1
Image_In	SELECT t.a1, t.a2 FROM table1 WHERE t.a1=1 AND t.a2 = 2
Image de tuple courant	Sans équivalence en SQL , les langages procédurales comme plsql , pgsql , transact sql sont capables.

Prochaine cours
SQL avancé et la Normalisation

- Projet sera publié demain
- TP1 est déjà publié

Questions en teams :

POSEZ DES QUESTIONS !

Bibliographie (1)

- **1. Cours IGE 487 et IFT 723 – Luc Lavoie et Christine Khnaisser**

http://info.usherbrooke.ca/llavoie/enseignement/IGE487/index.php

Modules

- **BD010_PRE**
- **BD010_SYN**

Excellents modules sur l'algèbre relationnelle.

Plusieurs modules sont copiés avec approbation de cet original, et enrichis avec des exemples.

Lire le Glossaire svp.

Bibliographie (2)

2. Ramez Elmasri , Shamkant Navathe – Fundamentals of database systems (7th edition)

Chapitre 8 sur l’algèbre relationnelle et les opérateurs.

Bibliographie (3)

3. Abraham Silberschatz Yale University, Henry F. Korth Lehigh University - DATABASE SYSTEM CONCEPTS 6th edition

Un livre qui met l’accent sur la pratique, en compétition avec (2).

Chapitre 2.6 , une présentation des opérateurs relationnels.