SMI6U05L : Bases de données
Le modèle relationnel (II cours)

Luigi Santocanale
LIS, Aix-Marseille Université

Contenu basé sur le cours de Rémi Eyraud

Plan

Autres opérations de l'algèbre relationnelle

Fonctions

Agrégats

Vers le langage SQL
- Interrogation des données
- Fonctions de type de données
- Agrégats

Plan

Autres opérations de l'algèbre relationnelle

Fonctions

Agrégats

Vers le langage SQL
- Interrogation des données
- Fonctions de type de données
- Agrégats

Autres opérations de l'algèbre relationnelle

- Opérations dérivées des opérations de base.
- Uniquement les principales, pas de liste exhaustive.
- Rappel des opérations de bases :
  - Union : `UNION(R, R')`
  - Différence : `EXCEPT(R, R')`
  - Produit cartésien : `TIMES(R,R')`
  - Projection : `PROJECT(R, Ensemble d'attributs)`
  - Restriction : `RESTRICT(R, Critère)`
  - Jointure : `JOIN(R, R', Critère)`

Intersection

- Def. L'intersection des relations R_1 et R_2 de mêmes schémas, est la relation R_3, toujours de même schéma, dont les tuples sont l'intersection de l'ensemble des tuples de R_1 et de l'ensemble des tuples de R_2.
- L'intersection est une opération commutative.
- Notations :
  - $R_1 \cap R_2$
  - INTERSECT(R_1, R_2)
  - AND(R_1, R_2)
- Redondance algébrique :

$$\begin{aligned} \text{INTERSECT}(R_1, R_2) &= \text{MINUS}(R_1, \text{MINUS}(R_1, R_2)) \\ &= \text{MINUS}(R_2, \text{MINUS}(R_2, R_1)) \end{aligned}$$

Intersection : exemple

Acteur1	Nom	Prénom	date de naissance
	Réno	Jean	NULL
	Debbouze	Jamel	18/06/75
	Benigni	Roberto	27/10/52

$\cap$

Acteur2	Nom	Prénom	date de naissance
	Réno	Jean	30/07/48
	Debbouze	Djamel	18/06/75
	Benigni	Roberto	27/10/52

$=$

Acteur1 $\cap$ Acteur2	Nom	Prénom	date de naissance
	Benigni	Roberto	27/10/52

Autres jointures

- La jointure est tellement essentielle à l'algèbre relationnelle que plusieurs opérations dérivées ont été introduites.
- Exemple déjà vu : la jointure naturelle
  (= jointure avec critère d'égalité entre les attributs de même nom, puis fusion des attributs de même nom).

Semi-jointure

- Le semi-jointure permet de ne garder les attributs que d'une des deux relations.
- On ne garde donc que les attributs de la première relation, mais les tuples ont été sélectionné par l'opération de jointure classique.
  C'est une jointure classique suivie d'un projection sur les attributs de la première relation.
- Notations :
  - `SEMI-JOIN(R, R', condition)`
  - $R \ltimes_{\texttt{critère}} R'$

Semi-jointure : exemple

Pays

nom	capitale	devise
France	Paris	3
Grèce	Athènes	3
Japon	Tokyo	42
Gabon	LibreVille	6

Monnaie

numéro	nom
3	Euro
1	Dollar
6	Franc CFA

JOIN(Pays, Monnaie, P.devise = M.numéro)

Pays.nom	capitale	devise	numéro	Monnaie.nom
France	Paris	3	3	Euro
Grèce	Athènes	3	3	Euro
Gabon	Libreville	6	6	Franc CFA

SEMI-JOIN(Pays, Monnaie, P.devise = M.numéro)

nom	capitale	devise
France	Paris	3
Grèce	Athènes	3
Gabon	Libreville	6

SEMI-JOIN(Monnaie, Pays, P.devise = M.numéro)

numéro	nom
3	Euro
6	Franc CFA

Jointure externe

- Contrairement à la jointure simple, la jointure externe n'élimine pas les tuples des relations concernées.
- Le résultat d'une jointure externe est celui d'une jointure simple union les tuples non concernés par la jointure (complétés par la valeur NULL).
- Notations :
  - `EXT-JOIN(`R_1`,` R_2`, critère)`
  - $R_1 \overset{\cdot}{\bowtie}_{\texttt{critère}} R_2$

Jointure externe : exemple

Pays	nom	capitale	monnaie
	Italie	Roma	3
	France	Paris	3
	Gabon	Libreville	6
	Bénin	Porto-Novo	6

Monnaie	num	nom
	1	Dollar US
	3	Euro
	6	Franc CFA

Pays P ⋈ **Monnaie M**

P.monnaie = M.num

P.nom	capitale	monnaie	num	M.nom
Italie	Roma	3	3	Euro
France	Paris	3	3	Euro
Gabon	Libreville	6	6	Franc CFA
Bénin	Porto-Novo	6	6	Franc CFA
NULL	NULL	NULL	1	Dollar US

Plan

Autres opérations de l'algèbre relationnelle

Fonctions

Agrégats

Vers le langage SQL
- Interrogation des données
- Fonctions de type de données
- Agrégats

Fonctions/expressions d'attributs

- Types de base : entier, réel, booléen, etc.
- On peut donc appliquer des fonctions aux valeurs d'attributs.
  Extension de l'algèbre relationnel [Zaniolo, 1985].
- On parle d'expressions évaluables d'attributs.
- Attributs de type arithmétique, opérateurs classiques.
  Utilisés pour
  - créer d'autres valeurs d'attribut comme argument de projection,
  - argument de projection de restrictions ou de jointures.
- La valeur NULL est absorbant :

$$1 + \mathit{NULL} = \mathit{NULL}, \qquad \mathit{NULL} - \mathit{NULL} = \mathit{NULL}.$$

Fonctions d'attributs (II)

Exemple :

```
Film(titre, réalisateur, année, nbprix, durée)
```

Durée est un entier correspondant au nombre de minutes.

- Récupérer en heures :

```
durée/60
```

- Compter les 20 minutes de pub :

```
(durée + 20)/60
```

Fonctions d'attributs (III)

- Visualiser la durée réelle d'une scéance :

```
PROJECT(Film, titre, durée + 20)
```

- Ce qui est valable sur les types arithmétiques l'est sur les autres types de base
  - chaînes de caractères : concaténation,
  - ...
- Chaque SGBDR fournit un ensemble de fonctions.

Plan

Autres opérations de l'algèbre relationnelle

Fonctions

Agrégats

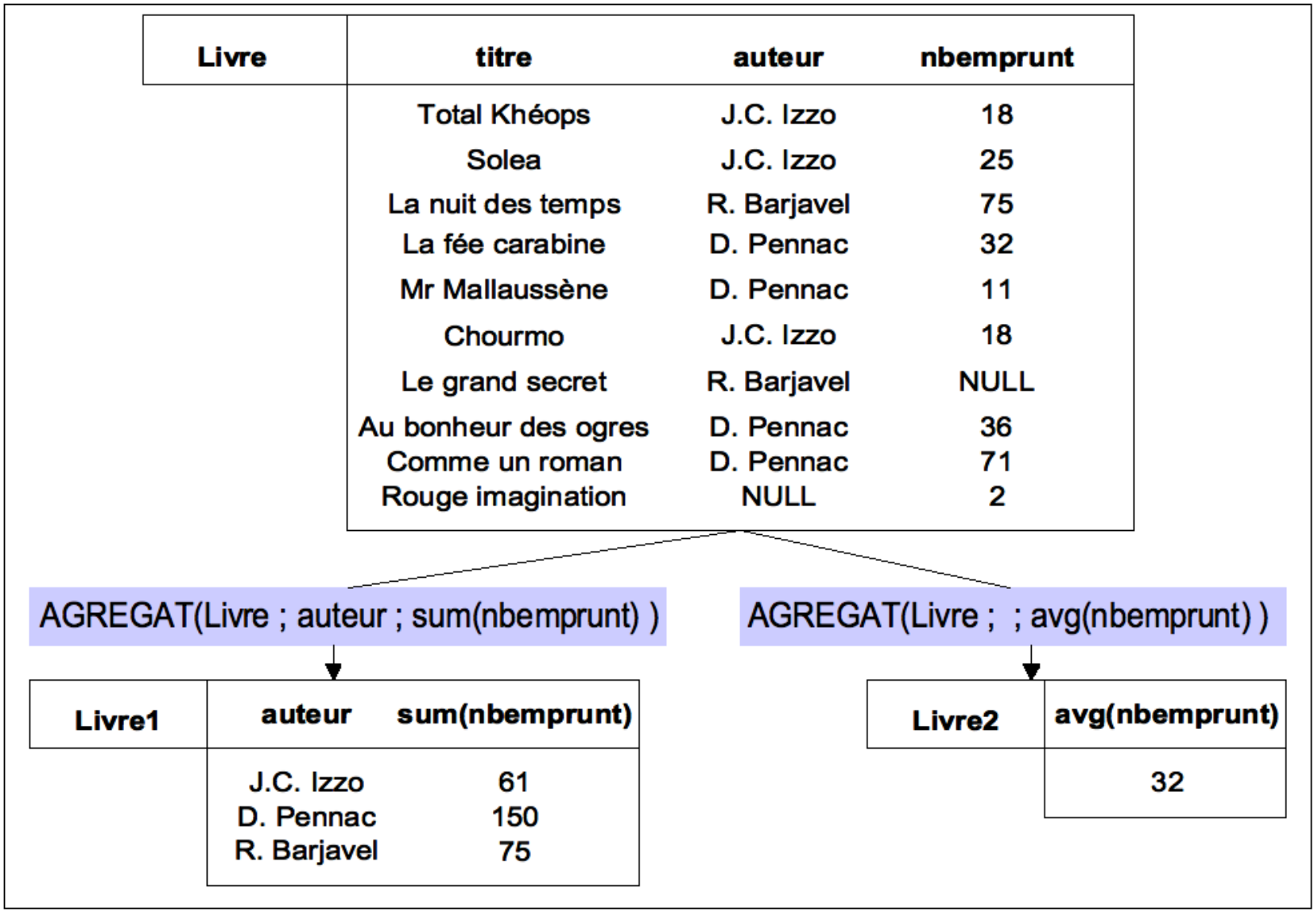
Vers le langage SQL
- Interrogation des données
- Fonctions de type de données
- Agrégats

Agrégats

- Notion essentielle mais pas facile d'accès.
- Jusqu'à présent :
  relation comme ensemble de tuples (i.e. de lignes).
- La notion d'agrégat permet de travailler sur une colonne d'une relation.
  Exemple : calculer la durée moyenne d'un films de la base.
- Supposons qu'on ait, dans une relation R, un attribut a, de type entier.
  On aimerait pouvoir :
  - calculer la somme des valeurs (de la colonne a) sur l'ensemble de tuples
  - récupérer le max des valeurs des tuples
  - compter le nombre d'éléments

```
SUM(a)
MAX(a)
COUNT(a)
```

Agrégats : exemples



Agrégats (suite)

- Notation :

S = AGREGAT(R; a; $f(b)$)

- On dit que
  - S est le résultat d'un calcul d'agrégat sur R,
  - via la fonction f appliquée à un attribut b,
  - où les tuples sont regroupés par égalité de valeur de l'attribut a.
- R aura 2 colonnes :
  - a, qui résumera l'ensemble des valeurs de l'attribut a dans la base,
  - la colonne $f(b)$, qui pour chaque valeur de a calculera la fonction f appliquée à b.
- C'est-à-dire

$$AGREGAT(R; a; f(b)) = \{ (x, f(Y_x)) \mid x \in \Pi_a(R), \quad Y_x = \{ y \mid (x, y) \in \Pi_{a,b}(R) \} \}.$$

Agrégat et expressions algébriques

- Note : les tuples ayant `NULL` pour valeur à l'attribut *b* ne sont pas pris en compte.
- Bien entendu, il est possible d'utiliser des agrégats dans les expression algébriques.
- Quelle est la durée moyenne des films réalisés par "Besson" ?

```
R1 = RESTRICT(Film, réalisateur='Besson')
RESULTAT = AGREGAT(R1 ; ; AVG(durée))
```

Plan

Autres opérations de l'algèbre relationnelle

Fonctions

Agrégats

Vers le langage SQL
- Interrogation des données
- Fonctions de type de données
- Agrégats

Des expressions algébriques à SQL

- Complétude relationnel de **SQL**.
- Clause de base :

```
SELECT ... FROM ... WHERE ...
```

  - select : projection
  - from : relation(s) sur laquelle porte la requête
  - where : restriction

- Dans cette partie : commandes de requête **SQL**.

Projection

- But : Obtenir l'ensemble des titres des films de la base.
- Expression relationnelle :

```
PROJECT(Film, titre)
```

- Expression **SQL** :

```
SELECT titre
FROM Film
```

SQL et modèle relationnel

- Attention : **SQL** $\neq$ modèle relationnel.
- Exemple : on peut avoir des lignes repetés dans une table **SQL** (d'une SGBDR).
- Pour éliminer les doublons :

```
SELECT DISTINCT ... FROM ... WHERE
```

- Considérez (pour le transparent suivant) la table suivante :

Film

titre	année	cinéaste
Cape Fear	1991	Scorsese
Cape Fear	1962	Thompson
Dumbo	2019	Burton
Dumbo	1941	NULL

DISTINCT, example I

- La requête

```
SELECT titre FROM Film
```

retourne

titre
Cape Fear
Cape Fear
Dumbo
Dumbo

DISTINCT, example II

- La requête

```
SELECT DISTINCT titre FROM Film
```

retourne

titre
Cape Fear
Dumbo

Restriction

- But :
  Obtenir l'ensemble des films (tous les attributs) dont la durée est supérieure à 120 minutes.
- Expression relationnelle :

```
RESTRICT(Film, duree > 120)
```

- Expression **SQL** :

```
SELECT *
FROM Film
WHERE duree > 120
```

Projection + restriction

- But :
  Obtenir l'ensemble des titres et durées des films durant plus de deux heures
- Expression relationnelle :

```
PROJECT(
    RESTRICT(Film, durée >120),
    titre, duree)
```

- En **SQL** :

```
SELECT titre, duree
FROM Film
WHERE duree > 120
```

Jointure

- 2 façons de réaliser une jointure : `JOIN` ou `produit cartésien` suivi d'une `restriction`.
- But : Obtenir l'ensemble des acteurs qui ont joué le personnage d'Astérix.
- Expression relationnelle :

```
R1 = NATURAL JOIN(Acteur, Casting)
R2 = RESTRICT(R1, personnage='Astérix')
RESULTAT = PROJECT(R2, nom, prénom)
```

Jointure (II)

- But :
  Obtenir l'ensemble des acteurs qui ont joué le personnage d'Astérix.
- En **SQL** (norme ANSI) :

```
SELECT nom, prénom
FROM
   Acteur JOIN Casting
        ON Casting.numacteur = Acteur.numacteur
WHERE personnage = 'Astérix'
```

- Note : `JOIN` suit directement `FROM`.
  Il est possible d'enchainer les `JOIN` en cascade.

Jointure (III)

- But :
  Obtenir l'ensemble des acteurs qui ont joué le personnage d'Astérix.
- En **SQL** (norme ANSI - commutativité) :

```
SELECT nom, prénom
FROM
    Casting JOIN Acteur
        ON Casting.numacteur = Acteur.numacteur
WHERE personnage = 'Astérix'
```

Jointure (IV)

- But :
  Obtenir l'ensemble des acteurs qui ont joué le personnage d'Astérix.
- En **SQL** (norme ANSI – traditionnelle, moins efficace) :

```
SELECT nom, prénom
FROM Casting, Acteur   -- produit cartésien
WHERE
    Casting.numacteur = Acteur.numacteur
    AND
    personnage = 'Astérix'
```

Jointure (naturelle)

- But :
  Obtenir l'ensemble des acteurs qui ont joué le personnage d'Astérix.
- En SQL (norme ANSI – mais à éviter) :

```
SELECT nom, prénom
FROM
Casting NATURAL JOIN Acteur
WHERE personnage = 'Astérix'
```

Jointure multiple

- Jointure entre plus de 2 relations.
- But :
  Obtenir les acteurs (nom, prénom) ayant joué dans 'la vie est belle' et, pour chacun, le personnage joué.
- Expression relationnelle :

```
PROJECT(
    RESTRICT(
        JOIN(Acteur,JOIN(Film,Casting)),
        titre = 'La vita è bella'),
    nom, prénom, personnage)
```

Jointure multiple

- But :
  Obtenir les acteurs (nom, prénom) ayant joué dans 'la vie est belle' et, pour chacun, le personnage joué.
- **SQL** (ANSI – avec JOIN) :

```
SELECT nom, prenom, personnage
FROM
Casting
JOIN Acteur ON Acteur.numacteur = Casting.numacteur
JOIN Film ON Film.numfilm = Casting.numfilm
WHERE titre = 'La vita è bella' ;
```

Fonctions de type de données

- SQL permet bien entendu d'utiliser les fonctions de type d'attributs.
- Sur la base filmographique :
  Obtenir le titre et la durée en heure de tous les films de plus de deux heures.
- **SQL** :

```
SELECT titre, duree/60
FROM Film
WHERE duree/60 >= 2
```

Fonctions de type de données

- Obtenir tous les acteurs (nom, prénom) nés en 1990.
- **SQL** :

```
SELECT nom, prenom
FROM acteur
WHERE datenaissance
BETWEEN DATE('1990-01-01') AND DATE('1990-12-31')
```

- **Résultat**

nom	prenom
Watson	Emma
Stewart	Kristen

Fonctions de type de données

- Obtenir tous les acteurs (nom, prénom) nés en 1990, classés par ordre alphabétique.
- **SQL** :

```
SELECT nom, prenom
FROM acteur
WHERE datenaissance
BETWEEN DATE('1990-01-01') AND DATE('1990-12-31')
ORDER BY nom
```

- **Résultat**

nom	prenom
Stewart	Kristen
Watson	Emma

Agrégats

- **SQL** permet d'utiliser les fonctions sur les type de données et les fonctions d'agrégat.
- Relation :

```
Livre(numlivre, titre, auteur, nbemprunt)
```

- But :
  Obtenir le nombre total d'emprunts pour chaque auteur.
- Expression relationnelle :

```
AGREGAT(Livre ; auteur ; sum(nbemprunt) )
```

Agrégats

- Relation :

```
Livre(numlivre, titre, auteur, nbemprunt)
```

- But :
  Obtenir le nombre d'emprunts pour chaque auteur.
- **SQL** :

```
SELECT auteur, SUM(nbemprunt)
FROM livre
GROUP BY auteur
```

Agrégats

- Relation :

```
Livre(numlivre, titre, auteur, nbemprunt)
```

- But :
  Obtenir la moyenne du nombre d'emprunts tous livres et tous auteurs confondus.
- Expression relationnelle :

```
AGREGAT(Livre ; ; avg(nbemprunt))
```

- **SQL** :

```
SELECT AVG(nbemprunt)
FROM Livre
```