



Bases de données | Informatique

Chapitre 4 – 04

Algèbre relationnelle

19 Mai 2021

Savoirs et compétences :

- ❑ BDD.C4 : Traduire dans le langage de l'algèbre relationnelle des requêtes écrites en langage courant
- ❑ BDD.C5 : Concevoir une base constituée de plusieurs tables, et utiliser les jointures symétriques pour effectuer des requêtes croisées
- ❑ BDD.S2 : Opérateurs usuels sur les ensembles dans un contexte de bases de données : union, intersection, différence.
- ❑ BDD.S3 : Opérateurs spécifiques de l'algèbre relationnelle : projection, sélection (ou restriction), renommage, jointure, produit et division cartésiennes ; fonctions d'agrégation : min, max, somme, moyenne, comptage.

Cours

1	Résumé des épisodes précédents	2
2	Problème	2
3	Structure du modèle relationnel	2
4	Algèbre relationnelle	3
4.1	Projection	3
4.2	Sélection	3
4.3	Renommage	4
4.4	Produit cartésien	4
4.5	Jointure simple	5
4.6	Union	5
4.7	Intersection	6
4.8	Différence	6
5	Mis sous le tapis	7
6	Agrégats	7
7	Conclusion	7

1 Résumé des épisodes précédents

On a déjà vu les notions suivantes.

1. MCD (Entité-Association) pour la représentation conceptuelle d'un problème.
2. MLD pour transcrire le MCD en tables.
3. Implantation dans une base de données SQL (MPD).
4. Les requêtes SQL.

2 Problème

Comment raisonner sur les opérations effectuées sur une base de données? Pour cela, on a besoin de modéliser ce problème correctement (comprendre : mathématiquement).

La modélisation que nous allons utiliser est **le modèle relationnel**.

On peut en distinguer deux parties.

Structure du modèle relationnel : modélisation des données (contenues dans des tables).

Algèbre relationnelle : modélisation mathématique des requêtes SQL.

3 Structure du modèle relationnel

On veut formaliser la notion de tables contenant des colonnes nommées.

Définition — Attribut, domaine. On appelle ensemble d'**attributs** un ensemble noté **att** (potentiellement infini).

Pour tout attribut $a \in \text{att}$, on appelle **domaine de a** un ensemble de **constantes** noté **dom**(a) (analogue au **type** de a).

Le domaine, noté **dom**, est l'union de toutes les constantes de tous les attributs :

$$\text{dom} = \bigcup_{a \in \text{att}} \text{dom}(a).$$

■ **Exemple** Attributs de notre base : titre, nom, prenom, id, date, datenaissance, idrealisateur, idfilm, idacteur, idpersonnage.

Le domaine de l'attribut titre : {"Gran Torino"; "The Good, the Bad and the Ugly"; "Study in Pink"; "Schindler's List"; "Dr Strangelove"; "Invictus"}.

Définition — Schémas. On appelle **schéma relationnel** un n -uplet d'attributs (on parlera aussi de champs d'un schéma relationnel). L'ensemble des noms des schémas relationnels, noté **relname**, est supposé disjoint de **att**.

Un **schéma de bases de données** est un ensemble fini de schémas relationnels.

■ **Exemple** Dans notre base de données, nous avons quatre schémas relationnels :

PERSONNE=(id,prenom,nom,datenaissance)

FILM=(id,titre,date,idrealisateur)

PERSONNAGE=(id,nom)

JOUE=(idacteur,idfilm,idrealisateur)

et nous considérons le schéma de base de donnée MPSIMDB=(PERSONNE,FILM,PERSONNAGE,JOUE).

Les champs de PERSONNE sont : id, prenom, nom, datenaissance.

On notera parfois FILM[id, titre, date, idrealisateur] pour dénoter FILM et rappeler les champs de FILM.

Pour deux n -uplets d'attributs U, V , on notera $V \subset U$ si les champs de V sont aussi dans U et s'y trouvent dans le même ordre.

■ **Exemple** On pourra noter $(\text{prenom}, \text{nom}) \subset (\text{id}, \text{prenom}, \text{nom}, \text{datenaissance})$ et l'on pourra aussi noter $\text{PERSONNE}[\text{prenom}, \text{nom}] = (\text{prenom}, \text{nom})$.

Définition — Relation. Une **relation** R (ou table) associée à un schéma relationnel $S = (A_1, \dots, A_n)$, ou **instance d'un schéma relationnel** $R[S]$, est un ensemble fini de n -uplets appartenant à $\text{dom}(A_1) \times \dots \times \text{dom}(A_n)$.

■ **Exemple** La table associée au schéma FILM[id, titre, date, idrealisateur] est la suivante.

(1, "Gran Torino"	, 2008, 3)
(2, "The Good, the Bad and the Ugly"	, 1966, 6)
(3, "Study in Pink"	, 2010, 7)
(4, "Schindler's List"	, 1993, 2)
(5, "Dr Strangelove"	, 1964, 1)
(6, "Invictus"	, 2009, 3)

Pour un élément t d'une relation R sur un schéma $S = (A_1, \dots, A_n)$, pour $T \subset S$ on notera $t[T]$ les éléments de t portant sur les champs de T .

■ **Exemple** Sur le schéma FILM[id, titre, date, idrealisateur] et la relation écrite précédemment, avec $t = (1, \text{"Gran Torino"}, 2008, 3)$, on pourra écrire $t[\text{id}] = 1$ et $t[\text{titre}, \text{date}] = (\text{"Gran Torino"}, 2008)$.

Définition — Base de données. Une **base de données** est la donnée d'un schéma de base de données et, pour chacun de ces schémas relationnels, d'une relation sur ce schéma.

■ **Exemple** La base de donnée MPSIMDB détaillée dans les cours précédent.

4 Algèbre relationnelle

On étudie des opérations sur les données d'une base (similaire aux LCI vue en cours de mathématiques). Nous détaillerons neuf opérations :

- | | | |
|----------------|--------------------------|------------------|
| 1. projection; | 4. produit cartésien; | 7. intersection; |
| 2. sélection; | 5. jointure (naturelle); | 8. différence. |
| 3. renommage; | 6. union; | |

4.1 Projection

Quels sont les noms et les prénoms des personnes de notre base de donnée? Pour répondre à la question, il suffit de prendre les colonnes nom et prenom de la table PERSONNE. On dit qu'on **projette** la table PERSONNE sur les attributs (nom, prenom).

Définition — Projection. Soit $n \in \mathbb{N}^*$ et $A_1, \dots, A_n \in \text{att}$. On appelle opération de **projection sur les attributs** (A_1, \dots, A_n) et l'on note π_{A_1, \dots, A_n} l'opération définie par $\pi_{A_1, \dots, A_n}(R) = \{t[A_1 \dots A_n] \mid t \in R\}$ pour toute relation R ayant au moins les attributs A_1, \dots, A_n .

Ainsi, la projection d'une relation sur (A_1, \dots, A_n) est une relation de schéma (A_1, \dots, A_n) .

En SQL, une projection se traduit par l'instruction SELECT (qui ne correspond donc pas à une sélection!).

■ **Exemple** On obtient les noms et les prénoms des personnes de notre base de donnée par l'opération $\pi_{\text{nom}, \text{prenom}}(\text{PERSONNE})$. La requête SQL traduisant cette projection est `SELECT nom, prenom FROM PERSONNE;`.

4.2 Sélection

Quelles sont les personnes dont le prénom est «Clint»? Pour répondre à la question, on **sélectionne**, dans la table PERSONNE, les nuplets dont le champ prenom est «Clint».

Définition — Sélection. Pour un critère de sélection C (fonction à valeurs booléennes définie sur un n -uplet de domaines), on définit l'opération de sélection σ_C qui, à toute relation R dont les champs sont compatibles avec C , associe $\sigma_C(R) = \{t \in R \mid C(t)\}$. Ainsi, on sélectionne les éléments de R vérifiant C , $\sigma_C(R)$ étant une relation de même schéma relationnel que R . Pour deux attributs $A, B \in \text{att}$ et $a \in \text{dom}$, on définit notamment les opérations

de sélection $\sigma_{A=a}$ et $\sigma_{A=B}$, comme les fonctions définies par

$$\sigma_{A=a}(R) = \{t \in R \mid t[A] = a\},$$

$$\sigma_{A=B}(R) = \{t \in R \mid t[A] = t[B]\}.$$

pour toute relation R ayant au moins A (resp. et B) comme attribut(s).

En SQL, la sélection se traduit par l'instruction WHERE.

■ **Exemple** Les personnes dont le prénom est «Clint» sont obtenues par $\sigma_{\text{prenom}=\text{"Clint"}}(\text{PERSONNE})$. La requête SQL traduisant cette sélection est

```
SELECT * FROM PERSONNE WHERE prenom = "Clint";
```

4.3 Renommage

Comment faire lorsque deux tables partagent un même attribut et que l'on veut les utiliser conjointement? On peut alors **renommer** un des champs concernés.

Définition — Renommage. Soit U un ensemble fini d'attributs. On appelle **renommage d'attributs** toute fonction $f : U \rightarrow \text{att}$ injective.

On appelle **opération de renommage** ρ_f associée à f l'opération qui, à $R[A_1, \dots, A_n]$ associe la relation

$$\rho_f(R)[f(A_1), \dots, f(A_n)] = \{t \mid t \in R\}.$$

Ainsi, ρ_f ne change que le schéma d'une relation, sans modifier ses éléments.

Souvent :

- U est clair d'après le contexte;
- et f laisse invariant tous les éléments de U sauf p éléments A_1, \dots, A_p dont les images respectives sont B_1, \dots, B_p .

l'opération de renommage ρ_f est alors notée $\rho_{A_1 \rightarrow B_1, \dots, A_p \rightarrow B_p}$

En SQL, une projection se traduit par l'instruction AS.

■ **Exemple** Renommer la colonne date de la table FILM en la colonne Date_de_sortie correspond à l'opération $\rho_{\text{date} \rightarrow \text{Date_de_sortie}}(\text{FILM})$. La requête SQL traduisant ce renommage est

```
SELECT id, titre, date AS Date_de_sortie, idrealisateur FROM FILM;
```

4.4 Produit cartésien

Peut-on obtenir une table comportant toutes les combinaisons possibles de couples d'éléments de PERSONNE et de JOUE?

Ⓡ En mathématiques, $A \times B$ désigne l'ensemble des couples (x, y) pour $x \in A$ et $y \in B$. Ici, ce sera l'ensemble des $x \oplus y$ où $x \oplus y$ désigne la concaténation des deux nuplets x et y , supposés n'avoir aucun attribut commun.

Définition — Produit cartésien. Soit R et S deux relations dont les ensembles de champs U et V vérifient $U \cap V = \emptyset$. On note $R \times S$ la relation portant sur les champs $U \cup V$ définie par $R \times S = \{u \oplus v \mid u \in R \text{ et } v \in S\}$.

On pourra bien entendu construire des produits cartésiens de plus de deux relations.

En SQL, on construit un produit cartésien en renseignant plusieurs tables, séparées par une virgule.

■ **Exemple** Le produit cartésien de PERSONNE et de JOUE se note tout simplement $\text{PERSONNE} \times \text{JOUE}$. La requête SQL traduisant ce produit est `SELECT * FROM PERSONNE, JOUE;`

Ⓡ Il faudra donc parfois renommer des colonnes avant de pouvoir construire des produits cartésiens. On pourra écrire en SQL `TABLE. attribut` afin de lever les ambiguïtés.

■ **Exemple** La requête SQL correspondant à l'opération $PERSONNE \times \rho_{id \rightarrow idfilm}(FILM)$ est

```
SELECT PERSONNE.id, nom, prenom, datenaissance,
       FILM.id AS idfilm, titre, date, idrealisateur
FROM PERSONNE, FILM;
```

4.5 Jointure simple

Quels sont les titres des films réalisés par des personnes dont le prénom est «Clint»?

Pour répondre :

1. on calcule $I = \sigma_{\text{prenom}="Clint"}(PERSONNE)$;
2. on calcule $J = \pi_{\text{titre}, \text{idrealisateur}}(FILM)$;
3. on calcule le produit $I \times J$;
4. on calcule la sélection $S = \sigma_{\text{id}=\text{idrealisateur}}(I \times J)$;
5. le résultat est $\pi_{\text{titre}}(S)$.

Les étapes 3 et 4 constituent un calcul de **jointure**.

Définition — Jointure. Soit R et S deux relations de champs U et V avec $U \cap V = \emptyset$, $A \in U$ et $B \in V$. Alors la **jointure symétrique de R et S selon (A, B)** est la relation notée $R[A=B]S$ de champ $U \cup V$, définie par

$$R[A=B]S = \sigma_{A=B}(R \times S).$$

La jointure :

- n'apporte **aucune expressivité** par rapport au produit suivi d'une sélection;
- en général, **se calcule plus facilement** (si on s'y prend bien).

■ **Exemple** Prenez un annuaire téléphonique de Lyon et la liste des enseignants de MPSI, calculez la jointure sur le couple nom de l'enseignant/nom de l'abonné :

- par produit puis sélection;
- directement.

En SQL, une jointure simple se traduit par l'instruction JOIN ON.

■ **Exemple** Pour obtenir les noms, prénoms des réalisateurs suivis des titres des films qu'ils ont réalisé, il suffit d'écrire l'opération $\pi_{\text{nom}, \text{prenom}, \text{titre}}(PERSONNE[id = idrealisateur]FILM)$.

La requête SQL traduisant cette jointure est

```
SELECT nom, prenom, titre
FROM PERSONNE JOIN FILM ON PERSONNE.id = idrealisateur;
```

4.6 Union

Quels sont les personnes dont le prénom est «Clint» ou «Martin»? Pour cela, on peut réaliser l'**union** des deux relations.

Définition — Union. Soit R et S deux relations de même schéma relationnel (**i.e.**, ayant les mêmes champs), alors l'**union** de R et de S est la relation $R \cup S = \{x \mid x \in R \text{ ou } x \in S\}$. C'est donc une relation de même schéma que R et S .

En SQL, une union se traduit par l'instruction UNION.

■ **Exemple** La table des personnes dont le prénom est «Clint» ou «Martin» s'obtient par l'opération.

$$\sigma_{\text{prenom}="Clint"}(PERSONNE) \cup \sigma_{\text{prenom}="Martin"}(PERSONNE)$$

La requête SQL traduisant cette union est

```
SELECT * FROM PERSONNE WHERE prenom = "Clint"
```

```
UNION
SELECT * FROM PERSONNE WHERE prenom = "Martin";
```

R On aurait pu remplacer l'union précédente par la sélection

$$\sigma_{\text{prenom}=\text{"Clint"} \text{ ou } \text{prenom}=\text{"Martin"}}(\text{PERSONNE}),$$

dont une traduction en SQL est

```
SELECT *
FROM PERSONNE
WHERE prenom = "Clint"
OR
prenom = "Martin";
```

4.7 Intersection

Quelles sont les personnes dont le prénom est «Clint» et le nom «Eastwood»? Pour cela, on peut réaliser l'**intersection** des deux relations.

Définition — Intersection. Soit R et S deux relations de même schéma relationnel (*i.e.*, ayant les mêmes champs), alors l'**intersection** de R et de S est la relation

$$R \cap S = \{x \mid x \in R \text{ et } x \in S\}.$$

C'est donc une relation de même schéma que R et S .

En SQL, une intersection se traduit par l'instruction INTERSECT.

■ **Exemple** La table des personnes dont le prénom est «Clint» et le nom «Eastwood» s'obtient par l'opération.

$$\sigma_{\text{prenom}=\text{"Clint"}}(\text{PERSONNE}) \cap \sigma_{\text{nom}=\text{"Eastwood"}}(\text{PERSONNE})$$

La requête SQL traduisant cette union est

```
SELECT * FROM PERSONNE WHERE prenom = "Clint"
INTERSECT
SELECT * FROM PERSONNE WHERE nom = "Eastwood";
```

R On aurait pu remplacer l'intersection précédente par la sélection $\sigma_{\text{prenom}=\text{"Clint"} \text{ et } \text{nom}=\text{"Eastwood"}}(\text{PERSONNE})$, dont une traduction en SQL est

```
SELECT * FROM PERSONNE WHERE prenom = "Clint" AND nom = "Eastwood";
```

On aurait aussi pu remplacer l'intersection précédente par la composition de sélections $\sigma_{\text{prenom}=\text{"Clint"}}(\sigma_{\text{nom}=\text{"Eastwood"}}(\text{PERSONNE}))$ dont une traduction en SQL est

```
SELECT * FROM (SELECT * FROM PERSONNE WHERE nom = "Eastwood") WHERE prenom = Clint;
```

4.8 Différence

Quelles sont les identifiants des personnes qui n'ont réalisé aucun film? Pour cela, on peut réaliser la **différence** des deux relations.

Définition — Différence. Soit R et S deux relations de même schéma relationnel (*i.e.*, ayant les mêmes champs), alors la **différence** de R et de S est la relation

$$R \setminus S = \{x \mid x \in R \text{ et } x \notin S\}.$$

C'est donc une relation de même schéma que R et S .

En SQL, une différence se traduit par l'instruction EXCEPT.

■ **Exemple** La table des identifiants des personnes n'ayant réalisé aucun film s'obtient par

$$\pi_{id}(PERSONNE) \setminus \rho_{idrealisateur \rightarrow id}(\pi_{idrealisateur}(FILM)).$$

La requête SQL traduisant cette union est

```
SELECT id FROM PERSONNE
EXCEPT
SELECT idrealisateur AS id FROM FILM;
```

««««< HEAD

5 Mis sous le tapis

En fait, SQL a quelques autres différences avec l'algèbre relationnelle :

- existence de requêtes agrégats en SQL;
- les résultats en SQL sont listes et non ensembles (utiliser l'instruction DISTINCT pour obtenir un ensemble à partir d'une liste).

6 Agrégats

On peut ajouter un opérateur d'agrégation à l'algèbre relationnelle.

Définition — fonction d'agrégation. Soit f une fonction prenant en argument une liste \mathcal{L} d'éléments de **dom**. On dit que f est une **fonction d'agrégation** si la valeur de $f(\mathcal{L})$ ne dépend pas de l'ordre des éléments de \mathcal{L} .

En pratique, on prendra pour fonctions d'agrégation :

- la fonction de comptage (de la longueur de la liste) notée count;
- max;
- min;
- la fonction moyenne arithmétique des éléments de la liste notée avg;
- la fonction somme des éléments de la liste notée sum.

Définition — opération d'agrégation. Soit A_1, \dots, A_n et B_1, \dots, B_p des attributs, R une relation dont le champ contient au moins tous ces attributs et f_1, \dots, f_p des fonctions d'agrégation. Alors on note $_{A_1, \dots, A_n} \gamma_{f_1(B_1), \dots, f_p(B_p)}(R)$ la relation obtenue :

- en regroupant les valeurs de R identiques sur les attributs A_1, \dots, A_n ;
- et en définissant de nouveaux attributs, notés $f_i(B_i)$, pour ces valeurs regroupées, pour tout $i \in \llbracket 1, p \rrbracket$, par application de la fonction d'agrégation f_i sur chacun de ces agrégats sur l'attribut B_i .

R Nous ne rentrerons pas dans le détail du schéma relationnel de cette relation.

■ **Exemple** Si l'on veut obtenir le nombre de films réalisés par chaque réalisateur (décrit par son identifiant), on utilise l'opération $_{idrealisateur} \gamma_{count(id)}(FILM)$. La requête SQL traduisant cette agrégation est

```
SELECT idrealisateur, COUNT(id)
FROM FILM
GROUP BY idrealisateur;
```

7 Conclusion

On a vu :

- algèbre relationnelle;
- (une partie de) SQL;
- le lien entre les deux.

TD

Sources :

Savoirs et compétences :

- ❑ *BDD.C4 : Traduire dans le langage de l'algèbre relationnelle des requêtes écrites en langage courant*
- ❑ *BDD.C5 : Concevoir une base constituée de plusieurs tables, et utiliser les jointures symétriques pour effectuer des requêtes croisées*
- ❑ *BDD.S2 : Opérateurs usuels sur les ensembles dans un contexte de bases de données : union, intersection, différence.*
- ❑ *BDD.S3 : Opérateurs spécifiques de l'algèbre relationnelle : projection, sélection (ou restriction), renommage, jointure, produit et division cartésiennes; fonctions d'agrégation : min, max, somme, moyenne, comptage.*