Chapitre $\underline{10001}_2$
Algèbre relationnelle

17 juillet 2018

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 3.0.1 (Attribut, domaine)**.** On appelle ensemble d'*attributs* un ensemble noté $\mathbf{att}$ (potentiellement infini).

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

Le domaine, noté $\mathbf{dom}$, est union de toutes les constantes de tous les attributs :

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

Exemple 3.0.2. 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 3.0.3 (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 $\mathbf{relname}$, est supposé disjoint de $\mathbf{att}$.

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

Exemple 3.0.4. Dans notre base de données, nous avons quatre schémas relationnels :

$$\begin{aligned} \texttt{PERSONNE} &= (\texttt{id}, \texttt{prenom}, \texttt{nom}, \texttt{datenaissance}) \\ \texttt{FILM} &= (\texttt{id}, \texttt{titre}, \texttt{date}, \texttt{idrealisateur}) \\ \texttt{PERSONNAGE} &= (\texttt{id}, \texttt{nom}) \\ \texttt{JOUE} &= (\texttt{idacteur}, \texttt{idfilm}, \texttt{idrealisateur}) \end{aligned}$$

et nous considérons le schéma de base de donnée

$$\texttt{MPSIMDB} = (\texttt{PERSONNE}, \texttt{FILM}, \texttt{PERSONNAGE}, \texttt{JOUE}).$$

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

On notera parfois $\texttt{FILM}[\texttt{id}, \texttt{titre}, \texttt{date}, \texttt{idrealisateur}]$ 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 3.0.5. On pourra noter

$$(\texttt{prenom}, \texttt{nom}) \subset (\texttt{id}, \texttt{prenom}, \texttt{nom}, \texttt{datenaissance})$$

et l'on pourra aussi noter

$$\texttt{PERSONNE}[\texttt{prenom}, \texttt{nom}] = (\texttt{prenom}, \texttt{nom}).$$

Définition 3.0.6 (Relation)**.** Une *relation* R (ou table) associée à un schéma relationnel $S = (A_1, \ldots, A_n)$, ou *instance d'un schéma relationnel* $R[S]$, est un ensemble fini de n-uplets appartenant à $\mathbf{dom}(A_1) \times \cdots \times \mathbf{dom}(A_n)$.

Exemple 3.0.7. La table associée au schéma

$$\texttt{FILM}[\texttt{id},\ \texttt{titre},\ \texttt{date},\ \texttt{idrealisateur}]$$

est la suivante.

$$\begin{aligned}
&(1, \texttt{"Gran Torino"} &&, 2008, 3)\\
&(2, \texttt{"The Good, the Bad and the Ugly"} &&, 1966, 6)\\
&(3, \texttt{"Study in Pink"} &&, 2010, 7)\\
&(4, \texttt{"Schindler's List"} &&, 1993, 2)\\
&(5, \texttt{"Dr Strangelove"} &&, 1964, 1)\\
&(6, \texttt{"Invictus"} &&, 2009, 3)
\end{aligned}$$

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

Exemple 3.0.8. Sur le schéma $\texttt{FILM}[\texttt{id},\ \texttt{titre},\ \texttt{date},\ \texttt{idrealisateur}]$ et la relation écrite précédemment, avec

$$t = (1, \texttt{"Gran Torino"}, 2008, 3),$$

on pourra écrire

$$t[\texttt{id}] = 1$$

et

$$t[\texttt{titre},\ \texttt{date}] = (\texttt{"Gran Torino"}, 2008).$$

Définition 3.0.9 (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 3.0.10. 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 ;
2. sélection ;
3. renommage ;
4. produit cartésien ;
5. division cartésienne ;
6. jointure (naturelle) ;
7. union ;
8. intersection ;
9. différence.

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 4.1.1 (Projection)**.** Soit $n \in \mathbb{N}^*$ et $A_1, \ldots, A_n \in \mathbf{att}$. On appelle opération de *projection sur les attributs* $(A_1, \ldots, A_n)$ et l'on note $\pi_{A_1,\ldots,A_n}$ l'opération définie par

$$\pi_{A_1,\ldots,A_n}(R) = \{\, t[A_1 \ \ldots \ A_n] \mid t \in R \,\}$$

pour toute relation R ayant au moins les attributs A_1, …, A_n.

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

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

Exemple 4.1.2. On obtient les noms et les prénoms des personnes de notre base de donnée par l'opération

$$\pi_{\texttt{nom, prenom}}(\texttt{PERSONNE}).$$

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 4.2.1 (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 élements de R vérifiant C, $\sigma_C(R)$ étant une relation de même schéma relationnel que R. Pour deux attributs $A, B \in \mathbf{att}$ et $a \in \mathbf{dom}$, 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 4.2.2. Les personnes dont le prénom est «Clint» sont obtenues par

$$\sigma_{\texttt{prenom="Clint"}}(\texttt{PERSONNE}).$$

La requête SQL traduisant cette sélection est

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

4.3 Renommage

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

Définition 4.3.1 (Renommage)**.** Soit U un ensemble fini d'attributs. On appelle *renommage d'attributs* toute fonction $f : U \to \mathbf{att}$ injective.

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

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

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 $A_1, \ldots, A_p$ dont les images respectives sont $B_1, \ldots, B_p$.

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

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

Exemple 4.3.2. Renommer la colonne `date` de la table `FILM` en la colonne `Date_de_sortie` correspond à l'opération

$$\rho_{\texttt{date} \to \texttt{Date\_de\_sortie}}(\texttt{FILM}).$$

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` ?

Remarque 4.4.1. 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 4.4.2 (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 4.4.3. Le produit cartésien de `PERSONNE` et de `JOUE` se note tout simplement `PERSONNE` $\times$ `JOUE`. La requête SQL traduisant ce produit est

```
SELECT *
FROM PERSONNE, JOUE;
```

Remarque 4.4.4. 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 ambiguités.

Exemple 4.4.5. La requête SQL correspondant à l'opération $\texttt{PERSONNE} \times \rho_{\texttt{id} \to \texttt{idfilm}}(\texttt{FILM})$ est

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

4.5 Division cartésienne

Quels sont les noms des acteurs ayant joué dans **tous** les films réalisés par Clint Eastwood ?

Considérons les notations suivantes

— $J[\texttt{nom}, \texttt{titre}]$ est l'ensemble des $(n, t) \in (\mathbf{dom}(\texttt{nom}), \mathbf{dom}(\texttt{titre}))$ pour chaque
  - n nom d'une personne ;
  - t titre de film ;
  - tels que n a joué dans t et Clint Eastwood a réalisé t.

— $E[\texttt{titre}]$ est l'ensemble des $t \in \mathbf{dom}(\texttt{titre})$ tels que t a été réalisé par Clint Eastwood.

— $K[\texttt{nom}]$ l'ensemble des $n \in \mathbf{dom}(\texttt{nom})$ tels que n a joué dans **tous** les films de Clint Eastwood.

K est la plus grande relation sur $\{\, \texttt{nom} \,\}$ vérifiant $K \times E \subset J$.
Cette relation est appelée *division cartésienne* de J par E.

Définition 4.5.1 (Division cartésienne)**.** Soit R et S deux relations, respectivement sur U et V avec $U \subset V$. Alors la *division cartésienne* de S par R est la relation sur $V \setminus U$ notée $S \div R$ et définie par :

$$\begin{aligned} S \div R &= \{\, t \in S[V \setminus U] \mid \forall u \in R, \quad t \oplus u \in S \,\} \\ &= \{\, t \mid \forall u \in R,\ \exists v \in S, \quad v[U] = u \text{ et } v[V \setminus U] = t \,\}. \end{aligned}$$

Comme nous le verrons plus tard, cette opération n'est pas traduite directement en SQL.

4.6 Jointure simple

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

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

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

Définition 4.6.1 (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 4.6.2. 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 4.6.3. Pour obtenir les noms, prénoms des réalisateurs suivis des titres des films qu'ils ont réalisé, il suffit d'écrire l'opération

$$\pi_{\texttt{nom},\ \texttt{prenom},\ \texttt{titre}}(\texttt{PERSONNE}[\texttt{id} = \texttt{idrealisateur}]\texttt{FILM}).$$

La requête SQL traduisant cette jointure est

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

4.7 Union

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

Définition 4.7.1 (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 4.7.2. La table des personnes dont le prénom est «Clint» ou «Martin» s'obtient par l'opération.

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

La requête SQL traduisant cette union est

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

Remarque 4.7.3. On aurait pu remplacer l'union précédente par la sélection

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

dont une traduction en SQL est

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

4.8 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 4.8.1 (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 4.8.2. La table des personnes dont le prénom est «Clint» et le nom «Eastwood» s'obtient par l'opération.

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

La requête SQL traduisant cette union est

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

Remarque 4.8.3. On aurait pu remplacer l'intersection précédente par la sélection

$$\sigma_{\texttt{prenom="Clint"}\text{ et }\texttt{nom="Eastwood"}}(\texttt{PERSONNE}),$$

dont une traduction en SQL est

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

Remarque 4.8.4. On aurait aussi pu remplacer l'intersection précédente par la composition de sélections

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

dont une traduction en SQL est

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

4.9 Différence

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

Définition 4.9.1 (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 4.9.2. La table des identifiants des personnes n'ayant réalisé aucun film s'obtient par

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

La requête SQL traduisant cette union est

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

5 Division cartésienne en SQL

On veut l'ensemble des noms des personnes ayant joué dans **tous** les films de Clint Eastwood.

Malheureusement, il n'y a pas d'opérateur de division dans SQL !

On peut calculer d'abord l'ensemble des noms n de personnes n'ayant pas joué dans **tous** les films de Clint Eastwood (réalisateur n° 3), c'est-à-dire pour n tel qu'il existe au moins un film de Clint Eastwood dans lequel n n'a pas joué, puis calculer le complémentaire par différence.

C'est (horriblement) compliqué :

```
SELECT nom FROM PERSONNE
EXCEPT
SELECT DISTINCT nom FROM (
    SELECT nom, titre FROM
        (SELECT nom FROM PERSONNE),
        (SELECT titre FROM FILM WHERE idrealisateur=3)
    EXCEPT
    SELECT nom, titre FROM PERSONNE, FILM, JOUE
        WHERE idfilm=FILM.id AND idacteur = PERSONNE.id
        AND idrealisateur=3
);
```

et encore, notre requête :

1. utilise le `id` de Clint Eastwood ;
2. est boguée en cas d'homonymes (on aurait dû faire des recherches sur les `id` et non les `nom`).

Deux façons de faire plus simple :

1. Faire autrement ;
2. Décomposer la requête grâce à des vues.

Faire autrement :

```
SELECT nom FROM PERSONNE, FILM, JOUE
WHERE idfilm=FILM.id
      AND
      idacteur = PERSONNE.id
      AND
      idrealisateur = 3
GROUP BY PERSONNE.id
HAVING COUNT(*) =
  (SELECT COUNT(*) FROM FILM WHERE idrealisateur=3)
```

— Résout le bogue précédent ;

— En introduit un autre si le diviseur est vide (i.e. si le réalisateur 3 n'a réalisé aucun film).

Décomposer la requête grâce à des vues.

```
-- table des id des films de Clint Eastwood
CREATE VIEW FILMS_EASTWOOD AS
SELECT FILM.id as idfilm
FROM FILM, PERSONNE
WHERE idrealisateur = PERSONNE.id
      AND
      nom = 'Eastwood'
      AND
      prenom = 'Clint';
```

(crée une table virtuelle qui contiendra le résultat de cette requête ; si le contenu des tables `FILM` et `PERSONNE` est modifié, la vue est modifiée)

```
-- table des couples acteurs/films
-- pour les films de Clint Eastwood
CREATE VIEW ACTEURS_EASTWOOD AS
SELECT id, nom, JOUE.idfilm
FROM FILMS_EASTWOOD, JOUE, PERSONNE
WHERE FILMS_EASTWOOD.idfilm = JOUE.idfilm
      AND
      idacteur=id;

-- produit cartésien PERSONNE * FILMS_EASTWOOD
CREATE VIEW PROD_PERSONNE_FILMS_EASTWOOD AS
SELECT id, nom, idfilm
FROM PERSONNE, FILMS_EASTWOOD;
```

Et finalement, la requête s'écrit

```
SELECT id, nom FROM personne
EXCEPT
SELECT id, nom FROM (
    SELECT * FROM PROD_PERSONNE_FILMS_EASTWOOD
    EXCEPT
    SELECT * FROM ACTEURS_EASTWOOD
);
```

(Ouf !)

6 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).

7 Agrégats

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

Définition 7.0.1 (fonction d'agrégation)**.** Soit f une fonction prenant en argument une liste $\mathscr{L}$ d'éléments de $\mathbf{dom}$. On dit que f est une *fonction d'agrégation* si la valeur de $f(\mathscr{L})$ ne dépend pas de l'ordre des éléments de $\mathscr{L}$.

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

— en regroupant les valeurs de R identiques sur les attributs $A_1, \ldots, A_n$;

— et en définissant de nouveaux attributs, notés $f_i(B_i)$, pour ces valeurs regroupées, pour tout $i \in [\![1, p]\!]$, par application de la fonction d'agrégation f_i sur chacun de ces agrégats sur l'attribut B_i.

Remarque 7.0.3. Nous ne rentrerons pas dans le détail du schéma relationnel de cette relation.

Exemple 7.0.4. Si l'on veut obtenir le nombre de films réalisés par chaque réalisateur (décrit par son identifiant), on utilise l'opération

$$_{\texttt{idrealisateur}}\gamma_{\text{count}(\texttt{id})}(\texttt{FILM}).$$

La requête SQL traduisant cette agrégation est

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

8 Conclusion

On a vu :

— algèbre relationnelle ;

— (une partie de) SQL ;

— le lien entre les deux.