Bases de Données : relations et requêtes

Stéphane Devismes

Université Grenoble Alpes

26 août 2020

Plan

Algèbre relationnelle

L'algèbre relationnelle relève des mathématiques, elle consiste à définir des opérations sur les relations.

Les principes de l'algèbre relationnelle sont utilisés par les systèmes de gestion des bases de données (SGBD), qui utilisent des langages comme SQL.

Relation

Soient $D_1, \ldots, D_n$ des ensembles. Une relation R sur $D_1, \ldots, D_n$ est un sous-ensemble du produit cartésien $D_1 \times \cdots \times D_n$:

$$R \subseteq D_1 \times \cdots \times D_n .$$

Exemple : Soient $E_1 = \{\text{Annette}, \text{Bernard}, \text{Cyril}\}$ et $E_2 = \{10, 12\}$ deux ensembles. Le produit cartésien $E_1 \times E_2$ est égal à

$$\{\langle \text{Annette},10\rangle, \langle \text{Annette}, 12\rangle, \langle \text{Bernard}, 10\rangle, \langle \text{Bernard}, 12\rangle, \langle \text{Cyril}, 10\rangle, \langle \text{Cyril}, 12\rangle\}$$

R_1, R_2 et R_3 définies ci-dessous sont des relations E_1 et E_2.

$$R_1 = \{\langle \text{Annette}, 10\rangle, \langle \text{Bernard}, 12\rangle, \langle \text{Cyril}, 10\rangle\}$$

$$R_2 = \{\langle \text{Annette},10\rangle, \langle \text{Annette}, 12\rangle, \langle \text{Bernard}, 10\rangle, \langle \text{Bernard}, 12\rangle, \langle \text{Cyril}, 10\rangle, \langle \text{Cyril}, 12\rangle\}$$

$$R_3 = \emptyset$$

n-uplets et attributs

Les éléments de la relation R sont des n-uplets (*e.g.*, doublet, triplet, *etc.*) $\langle x_1, \ldots, x_n \rangle$ avec $x_i \in D_i$ pour chaque i.

Pour chaque i de 1 à n on choisit un nom A_i appelé attribut, pour nommer le i-ème élément de chaque n-uplet de R : x_i est la valeur de l'attribut A_i dans le n-uplet $\langle x_1, \ldots, x_n \rangle$.

Par exemple, pour $R_1 = \{\langle \text{Annette}, 10 \rangle, \langle \text{Bernard}, 12 \rangle, \langle \text{Cyril}, 10 \rangle\}$, on choisit les attributs prénom et âge.

La valeur de l'attribut prénom dans le doublet $\langle \text{Annette}, 10 \rangle$ est Annette.

Une relation sur $D_1, \ldots, D_n$ est représentée en `SQL` par une table. Cette table a n colonnes, la colonne i a pour nom l'attribut A_i et elle contient des éléments de D_i. La table a une ligne pour chaque n-uplet $\langle x_1, \ldots, x_n \rangle$.

Par exemple, pour $R_1 = \{\langle \text{Annette}, 10 \rangle, \langle \text{Bernard}, 12 \rangle, \langle \text{Cyril}, 10 \rangle\}$, on a la table de trois lignes et deux colonnes suivantes :

prénom	âge
Annette	10
Bernard	12
Cyril	10

Relation *vs* table

Une relation est un ensemble de n-uplets, donc

- **il ne peut pas y avoir deux n-uplets égaux** et
- **il n'y a pas d'ordre sur les n-uplets**.

Par contre une table est une liste de lignes, donc

- **une table peut comporter plusieurs lignes égales**, et
- **les lignes d'une table sont dans un certain ordre**.

Cependant, on peut enlever les doublons dans une table, et on peut imposer un ordre précis pour l'affichage des n-uplets.

Mais **ces opérations sont coûteuses** (le tri est en $\Theta(n \log n)$), il ne faut les utiliser que lorsque c'est nécessaire.

Type

Dans une relation sur $D_1, \ldots, D_n$ avec comme attributs $A_1, \ldots, A_n$, l'ensemble D_i est le domaine ou le type de l'attribut A_i.

On se restreint à trois possibilités pour les domaines : nombres, chaînes de caractères ou dates, qu'on notera respectivement `Nombre`, `Chaine`, `Date`.

Les types en `SQL` sont plus précis : il y a plusieurs types de nombres et de chaînes de caractères, afin de préciser la place maximale nécessaire pour stocker leurs éléments en mémoire.

ATTENTION : en `SQL`, **une chaîne de caractères s'écrit avec des quotes simples**, par exemple 'toto'.

Identifiant

Un identifiant ou une clé d'une relation R est un ensemble K d'attributs de R tel que les valeurs de ces attributs **déterminent (de manière unique)** les valeurs de tous les attributs.

Autrement dit, si deux lignes ont le même identifiant, alors elles sont égales.

De plus, cet ensemble d'attributs doît être minimal :

- ou bien K est formé d'un seul attribut,
- ou bien K est formé de plusieurs attributs et dès qu'on en enlève un la propriété d'identifiant n'est plus vérifiée.

Il peut y avoir plusieurs identifiants pour une relation, dans ce cas on en choisit un qu'on appelle identifiant primaire ou clé primaire.

Remarque. Très souvent, une relation a un seul identifiant et il est formé d'un seul attribut.

Vocabulaire : algèbre relationnelle *vs* SQL

Le vocabulaire de SQL est différent du vocabulaire de l'algèbre relationnelle, selon le tableau ci-dessous. En pratique, ces couples de mots fonctionnent comme des synonymes.

Algèbre relationnelle	SQL
relation	table
attribut	nom de colonne
n-uplet	ligne
domaine	type
identifiant	clé

Schéma d'une relation

Le schéma d'une relation est formé de :

- le nom de la relation,
- la liste des attributs avec leurs domaines,
- l'identifiant (ou les identifiants) de la relation.

On note $R(A_1 : D_1, \ldots, A_n : D_n)$, ou seulement $R(A_1, \ldots, A_n)$ lorsque les domaines sont connus par ailleurs, et on souligne d'un seul trait l'ensemble d'attributs qui forme l'identifiant primaire.

Spécification d'une relation

La spécification d'une relation est une propriété qui caractérise, parmi tous les n-uplets de $D_1 \times \cdots \times D_n$, ceux qui sont dans la relation.

Toute relation intéressante a une spécification, et il faut la donner. On la note :

- $\langle x_1, \ldots, x_n \rangle \in R \iff$
  ... une description de la propriété qui caractérise R, en français et en utilisant les symboles $x_1, \ldots, x_n$.

Remarque

Une **relation** est définie par son schéma et sa spécification, qui ne dépendent pas des valeurs précises contenues dans la base de données.

De plus, une relation a une valeur, qui dépend des valeurs contenues dans la base de données.

Requêtes

Une base de données est constituée de (une ou) plusieurs relations.

Pour avoir des informations sur le contenu d'une base de données on pose des questions qui s'appellent des requêtes. Les requêtes n'utilisent que le schéma des relations, jamais leur valeur.

Une requête construit une relation, à partir des relations de la base, en utilisant des opérations sur les relations.

Chaque opération part d'une ou de plusieurs relations et calcule une nouvelle relation.

Requêtes en algèbre relationnelle et requêtes en SQL

Dans ce cours, nous allons écrire des requêtes en deux formes distincts : relationnelle (arborescente) et SQL.

Pourquoi ?

- Nous étudirons le SQL d'Oracle, or il existe de nombreux autres SQL avec des petites différences (Oracle ne suit même pas la norme SQL2 !).
- Il existe des éditeurs SQL graphiques (*i.e.*, QBE, *Query By Example*).
- L'algèbre relationnelle des mathématiciens ne correspond pas exactement aux opérations fournies par SQL.

  (Cependant, dans ce cours nous proposons une variante de l'algèbre relationnelle qui est mieux adaptée à SQL.)

Remarque : Il existe une représentation de l'algèbre relationnelle, plus traditionnelle, en ligne (en 1D). Mais, elle est plus difficile à lire.

Une école

Une école souhaite créer une base de données pour enregistrer et tenir à jour des informations sur ses élèves : **nom, prénom, date de naissance, option, notes**.

On fait l'hypothèse, pour simplifier, qu'**il ne peut pas y avoir plusieurs élèves de même nom et prénom**.

Ces informations forment une relation Eleves.

Les élèves

- Le schéma de la relation `Eleves` :

  `Eleves(`<u>`nom : Chaine, prénom : Chaine`</u>`, naissance : Date, opt : Nombre)`

  regroupe les informations suivantes :
  - Nom de la relation : `Eleves`.
  - Attributs (avec leur domaine) : `nom, prénom : Chaine` ; `naissance : Date` ; `opt : Nombre`.
  - Identifiant : (`nom,prénom`)
- La spécification de la relation `Eleves` précise sa sémantique :

  $\langle n, p, j, o\rangle \in$ `Eleves` $\iff$ *l'élève de nom n et de prénom p est né le jour j et a choisi l'option numéro o.*

Les activités

Plusieurs activités sont proposées aux élèves. Elles sont répertoriées dans une relation `Activites`. Le type d'une activité peut être par exemple sport, musique, ...

- Schéma de la relation `Activites` :

  `Activites(`<u>`activite : Chaine`</u>`, typeAct : Chaine)`
- Spécification de la relation `Activites` :

  $\langle a, t\rangle \in$ `Activites` $\iff$ *l'activité a est proposée aux élèves, elle est de type t.*

Inscriptions

Les inscriptions des élèves aux activités forment une relation `Inscriptions`.

- Schéma de la relation `Inscriptions` :

  `Inscriptions(`<u>`nom : Chaine, prénom : Chaine, activite : Chaine`</u>`)`
- Spécification de la relation `Inscriptions` :

  $\langle n, p, a\rangle \in$ `Inscriptions` $\iff$ *l'élève de nom n et de prénom p est inscrit à l'activité a.*

Maths

Le professeur de mathématiques reporte les notes des élèves dans une relation `Maths`.

- Schéma de la relation `Maths` :

  `Maths(`<u>`nom : Chaine, prénom : Chaine`</u>`, note : Nombre)`
- Spécification de la relation `Maths` :

  $\langle n, p, v\rangle \in$ `Maths` $\iff$ *l'élève de nom n et de prénom p a la note v en mathématiques.*

Bio

De même, le professeur de biologie reporte les notes des élèves dans une relation `Bio`.

- Schéma de la relation `Bio` :

  `Bio`(`nom : Chaine`, `prénom : Chaine`, `note : Nombre`)

- Spécification de la relation `Bio` :

  $\langle n, p, v \rangle \in$ `Bio` $\iff$ *l'élève de nom n et de prénom p a la note v en biologie.*

Histoire

Le cours d'histoire est facultatif. Le professeur d'histoire reporte les notes des élèves qui suivent son cours dans une relation `Histoire`.

- Schéma de la relation `Histoire` :

  `Histoire`(`nom : Chaine`, `prénom : Chaine`, `note : Nombre`)

- Spécification de la relation `Histoire` :

  $\langle n, p, v \rangle \in$ `Histoire` $\iff$ *l'élève de nom n et de prénom p suit le cours d'histoire, où il a la note v.*

Table Eleves

Eleves			
nom	prenom	naissance	opt
Arcila	Elise	15-MAY-89	1
Baume	Michel	07-JAN-89	1
Corse	Helene	30-NOV-87	2
Diaz	Juana	06-APR-88	2
Hassan	Franck	25-JAN-89	1
Leblanc	Michel	03-SEP-80	2
Tayachi	Gabriel	02-JUN-88	1

Tables Activites et Inscription

Activites	
activite	typeAct
batterie	musique
guitare	musique
harmonica	musique
piano	musique
violon	musique
athletisme	sport
foot	sport
ski	sport

Inscriptions		
nom	prenom	activite
Arcila	Elise	ski
Baume	Michel	harmonica
Hassan	Franck	foot
Leblanc	Michel	foot
Leblanc	Michel	piano
Tayachi	Gabriel	guitare
Tayachi	Gabriel	ski

Tables Maths, Bio et Histoire

Maths		
nom	prenom	note
Arcila	Elise	19
Baume	Michel	10
Corse	Helene	7
Diaz	Juana	13
Hassan	Franck	15
Leblanc	Michel	19
Tayachi	Gabriel	8

Bio		
nom	prenom	note
Arcila	Elise	17
Baume	Michel	15
Corse	Helene	9
Diaz	Juana	18
Hassan	Franck	15
Leblanc	Michel	15
Tayachi	Gabriel	18

Histoire		
nom	prenom	note
Baume	Michel	16
Corse	Helene	19
Leblanc	Michel	12

Projection (ou sélection) en SQL

```
SELECT attribut1, attribut2, attribut3  FROM table1
```

```
SELECT * FROM table1
```

(sélectionne toutes les colonnes de la table table1)

```
SELECT DISTINCT attribut1, attribut2, attribut3
FROM table1
```

(enlève les doublons)

```
ORDER BY attribut1, attribut2
```

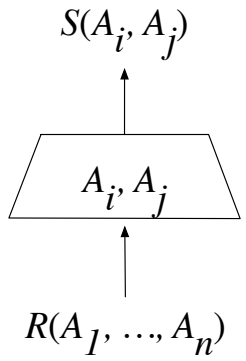
(ordonne sans enlever les doublons pour l'affichage)
(par défaut, ordre croissant : `ASC`)

Ordre décroissant :

```
ORDER BY attribut1, attribut2 DESC
```


Projection (ou sélection)

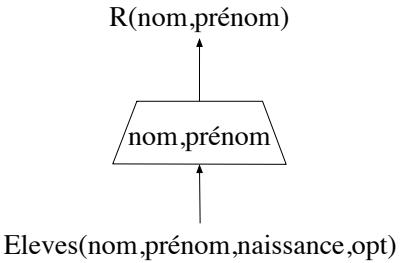
Conserver seulement les colonnes d'attributs A_i, A_j.



Projection (ou sélection) : exemples en AR (1/3)

Quels sont les élèves de l'école (leurs nom et prénom) ?

- Schéma : R(nom : Chaine, prénom : Chaine)
- Spécification : $\langle n, p \rangle \in \mathtt{R} \iff$ *l'élève de nom n et de prénom p fait partie de l'école.*
- Algèbre relationnelle :



Projection (ou sélection) : exemples en SQL (1/3)

Quels sont les élèves de l'école (leurs nom et prénom) ?

```
SELECT nom, prenom
FROM Eleves;
```

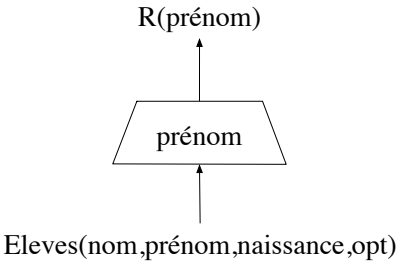
ou bien, pour un résultat ordonné :

```
SELECT nom, prenom
FROM Eleves
ORDER BY nom, prenom;
```


Projection (ou sélection) : exemples en AR (2/3)

Quels sont les prénoms des élèves de l'école ?

- Schéma : R(prénom : Chaine)
- Spécification : $\langle p \rangle \in R \iff$ *il y a un élève de prénom p dans l'école.*
- Algèbre relationnelle :



Projection (ou sélection) : exemples en SQL (2/3)

Quels sont les prénoms des élèves de l'école ?

```
SELECT DISTINCT prenom
FROM Eleves ;
```

attention, la requête suivante fournit un résultat avec répétition :

```
SELECT prenom
FROM Eleves ;
```


Projection (ou sélection) : exemples en SQL (3/3)

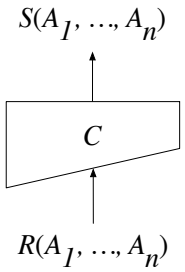
Quels sont les prénoms des élèves de l'école (ordonnée par option puis par ordre alphabétique) ?

Rappel : pas d'ordre en AR

```
SELECT DISTINCT prenom
FROM Eleves
ORDER BY opt, nom, prenom;
```

Restriction

Conserver seulement les lignes vérifiant la condition C.

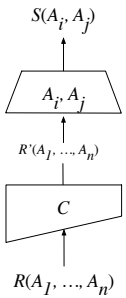


Remarque : on ne peut pas écrire une restriction seule en `SQL`.

Décomposition

Une requête est composée d'opérations (éventuellement une seule).

Dans la représentation graphique, on peut faire apparaître les relations intermédiaires, par exemple ici :

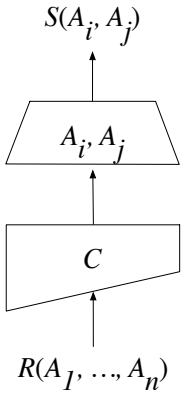


où $R'(A_1, \ldots, A_n)$ est formée des lignes de $R(A_1, \ldots, A_n)$ vérifiant la condition C.

Restriction puis projection

C'est la forme de requête « fondamentale » en SQL.

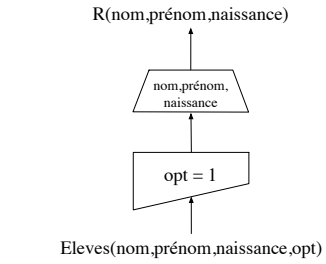
Afficher les colonnes d'attributs A_i, A_j des lignes vérifiant la condition C.



Restriction : exemples en AR

Quelles sont les noms, prénoms et dates de naissance des élèves de l'option 1 ?

- Schéma : `R(` <u>`nom : Chaine, prénom : Chaine`</u>`, naissance : Date)`
- Spécification : $\langle n, p, j \rangle \in$ `R` $\iff$ *l'élève de nom n et de prénom p est dans l'option 1 et il est né le jour j.*
- Algèbre relationnelle :



Restriction : exemples en SQL

Quelles sont les noms, prénoms et dates de naissance des élèves de l'option 1 ?

```
SELECT nom, prenom, naissance
FROM Eleves
WHERE opt = 1;
```


Conditions

La restriction (WHERE) est relative à une condition donnée C.

- Comparaisons (sensées !) entre des expressions construites à partir des attributs et des constantes, utilisant les opérateurs de comparaison usuels $=, <>, <, <=, >, >=, \ldots$ et aussi BETWEEN (inclus dans un intervalle, au sens large) ou encore IN (appartient à un ensemble).

  $A_1 = 5$, $A_1 =$ 'Blah', $A_1 =$ '15 janvier 2008', $A_1 > 0$, $A_1 + A_2 + 3 = A_3$, ...

- Recherches de motifs : « de la forme » (LIKE) : utilisation de jokers (wildcards) pour le pattern matching (% et _) :

  A_1 LIKE '%*bl*%', A_1 LIKE '*bl*__'

  % désigne une chaîne quelconque, éventuellement vide, et _ désigne un caractère. **LIKE est sensible à la casse**.

Exemples de conditions en SQL

Quels sont les élèves de prénom Michel ?

Quels sont les élèves dont le nom ne contient pas c ?

Conditions complexes

Les conditions peuvent être combinées en utilisant les connecteurs logiques usuels

- "et" ($\wedge$, AND),
- "ou" ($\vee$, OR),
- "non" ($\neg$, NOT),

avec leur signification usuelle. En particulier, "ou" n'est pas exclusif.

Conditions complexes : exemples en SQL

Quels sont les élèves ayant pour prénom Michel ou Gabriel ?

```
SELECT nom, prenom FROM Eleves
WHERE prenom = 'Michel' OR prenom = 'Gabriel';

-- ou bien
SELECT nom, prenom FROM Eleves
WHERE prenom IN ('Michel', 'Gabriel');

-- ATTENTION!!! mais pas
SELECT nom, prenom FROM Eleves
WHERE prenom = 'Michel' OR 'Gabriel';

-- ERROR ... invalid relational operator
```


Les dates en SQL

Le standard SQL2 préconise un type DATE pour année, mois et jour, et un type TIME pour heure, minute, seconde.

Mais en Oracle tout cela est regroupé dans le type DATE !

Opérations sur les dates en SQL (1/2)

Le système a un format interne pour les dates, qu'on n'utilise pas directement.

- Pour être imprimée, une date doit être convertie en chaîne.

  Par défaut Oracle utilise le format DD-MON-YY, par exemple 01-APR-98.

  On peut imposer un autre choix en utilisant TO_CHAR, par exemple, si d est de type DATE, alors TO_CHAR(d, 'YYYY/MM/DD') s'imprime 1998/04/01.

- Pour convertir une chaîne en date, on utilise soit le format par défaut soit TO_DATE

  Par exemple TO_DATE('1998/05/31:12:00:00AM', 'yyyy/mm/dd:hh:mi:ssam')

- La fonction SYSDATE retourne la date courante.

Opérations sur les dates en SQL (2/2)

- On peut comparer des dates en utilisant $=$, $!=$, $>$, *etc.*
- On peut soustraire deux dates, on obtient un nombre flottant qui exprime la différence entre les deux dates en jours.

  On peut aussi ajouter ou soustraire un nombre à une date, ce nombre est considéré comme un nombre de jours. Par exemple, SYSDATE+1 est demain au même moment.

- On peut tronquer une date en utilisant la fonction TRUNC.

  Par exemple :
  TRUNC(TO_DATE('22-AUG-03'), 'YEAR') retourne '01-JAN-03'
  TRUNC(TO_DATE('22-AUG-03'), 'MONTH') retourne '01-AUG-03'
  TRUNC(TO_DATE('22-AUG-03'), 'DDD') retourne '22-AUG-03'
  Attention, DAY désigne le premier jour de la semaine !
  TRUNC(TO_DATE('22-AUG-03'), 'DAY') retourne '17-AUG-03'

Opérations sur les dates en SQL : exemples (1/2)

Quels sont les élèves nés le 25 janvier 1989 ?

```
SELECT nom, prenom FROM Eleves
WHERE TO_CHAR(naissance,'DD-MON-YY') = '25-JAN-89';
```

- - ou bien

```
SELECT nom, prenom FROM Eleves
WHERE TO_CHAR(naissance,'DD-MM-YYYY') = '25-01-1989';
```

- - ou bien

```
SELECT nom, prenom FROM Eleves
WHERE naissance = TO_DATE('25-01-1989','DD-MM-YYYY');
```


Opérations sur les dates en SQL : exemples (2/2)

Quels sont les élèves nés en 1989 ?

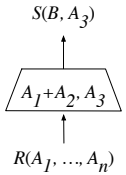
```
SELECT nom, prenom FROM Eleves
WHERE TO_CHAR(naissance,'YY') = '89';
```

- - ou bien

```
SELECT nom, prenom FROM Eleves
WHERE TO_CHAR(naissance,'YYYY') = '1989';
```


Opérations « horizontales » (par colonnes)

Calculer la somme des colonnes A_1 et A_2 en la renommant B, et conserver la colonne A_3.



Plus généralement, cela permet d'effectuer la même opération sur toutes les lignes.

Bien entendu il faut que les opérations soient compatibles avec les domaines des attributs.

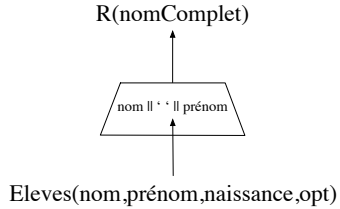
- $+, -, *, /$ pour les opérations de base sur deux colonnes de nombres,
- $||$ pour concaténer deux colonnes de chaînes.

Attention aux priorités sur les opérateurs, voir la documentation, utiliser des parenthèses. . .

Opérations « horizontales » : exemples en AR

Donner le prénom et le nom des élèves dans un seul attribut

- Schéma : R(nomComplet : Chaine)
- Spécification : $\langle nc \rangle \in$ R $\iff$ *nc est le concaténé du prénom et du nom d'un élève.*
- Algèbre relationnelle :



Opérations « horizontales » : exemples en SQL

Donner le prénom et le nom des élèves dans un seul attribut

```
SELECT prenom || ' ' || nom AS nomComplet
FROM Eleves;
```