Informatique IV

Initiation aux bases de données

I Introduction aux bases de données

Introduction

Les masses de données à l'heure actuelle :

- Livre ~ 1 Million de caractères ~ 1 Mo
- BNF ~ 10 Millions de livres ~ 10 Teraoctets
- Base de données type (assurance, banque, site web marchand, etc..) ~ pétaoctet ~ 100 BNF
- Tweeter : ~ 1 BNF par Jour [Le congrès américain a décidé de sauvegarder ce patrimoine !!]
- Web actuel : ~ 1 Milliard de BNF

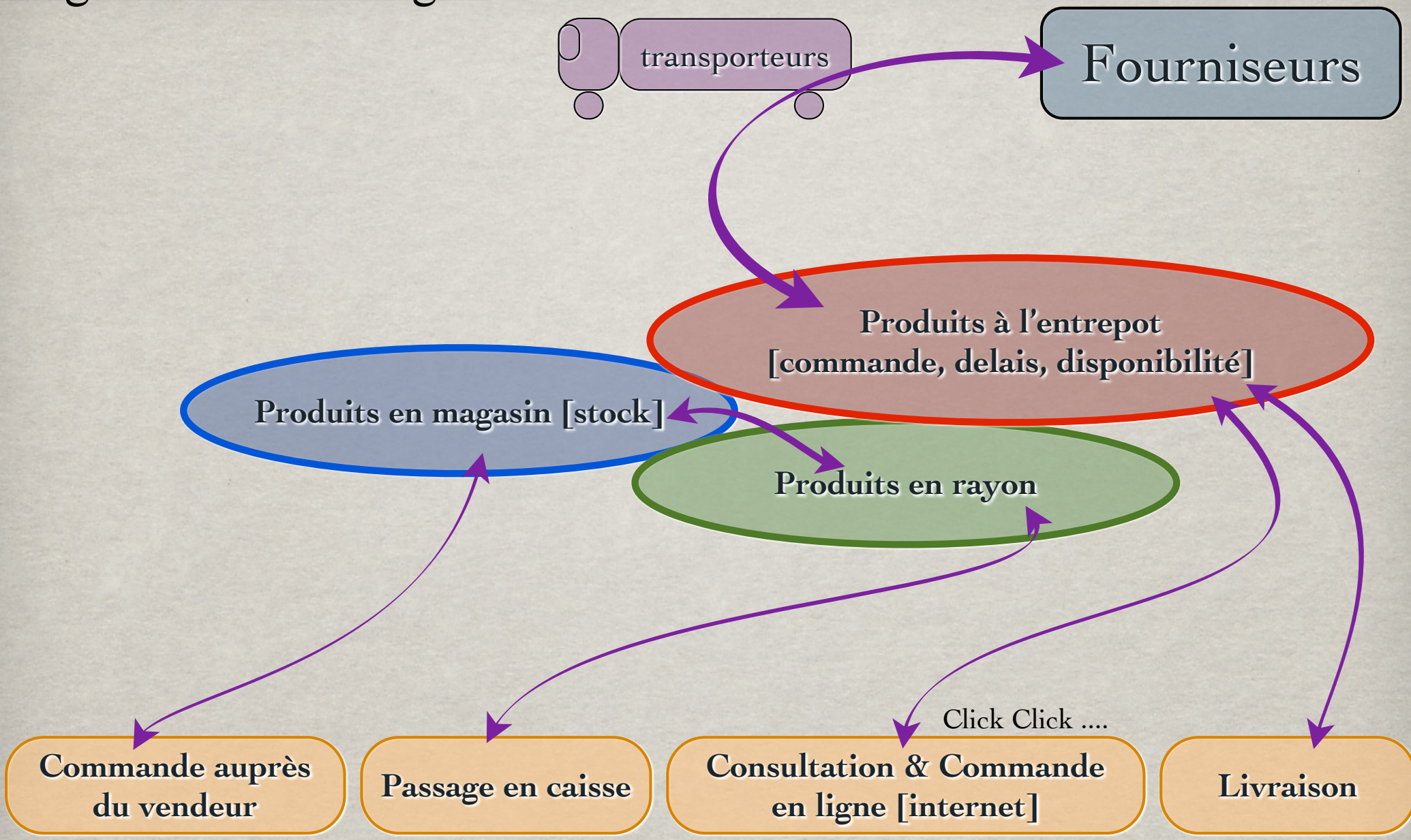
Les bases de données sont omniprésentes dans tous les secteurs économiques.

Objectifs :

- Présenter la réalité concrète d'une base de données.
- se familiariser avec ces objets très simples & structurés [BD relationnelles]
  comprendre le formalisme abstrait sous-jacent [algèbre relationnelle]

- savoir formuler des requêtes SQL
- créer / manipuler une base de données simple [TP]

Magasin de bricolage :



- Toutes ces opérations quotidiennes sont impossibles sans une base de données dynamiques

- La base de donnée est mise à jour en permanence par tous les acteurs économiques.

Exemples simples de mise à jour des tables constituants une base de données :

- chaque produit entrant doit être rentré dans une table «Produits» [INSERT]

- Pour chaque produit il y a un certain nombre d'articles disponibles «Article»

- lors d'un achat il faut :

  - Eliminer l'article de sa table [DROP], et modifier la disponibilité des articles [UPDATE]

  - Ajouter la vente dans la table des «Transactions» [INSERT]
    mais également enregistrer l'achat dans la table «Clients» [INSERT].

- Lors d'une consultation internet il faut pouvoir sélectionner tous les produits répondant à certains critères simples (prix, marque, couleur, disponibilité, etc) [**SELECT**]

- Lorsqu'un vendeur vérifie la disponibilité en magasin en stock : [**SELECT**]

On verra ainsi qu'une Base De Données est un ensemble de tables :

- «Produits» toutes les données sur les produits
- «Clients» toutes les données sur les clients.
- etc ....

Vision tabulaire du relationnel : BD <=> ensemble de Table et des relations entre elles.

Toutes ces opérations sont réalisées sur des interfaces graphiques très simples, mais sont transcrites par un **SGBD** [Système de Gestion de Base de Données] dans un **langage structuré de requêtes**, qui agit sur les tables d'une base de données.

SQL : Structured Query Language

C'est un language construit sur la base de l'algèbre relationnelle inventée à dessein par **IBM** en 1970.

Il permet d'effectuer des opérations très similaires à celles de la théorie des ensembles, sur de tables de données et d'établir des relations entre ces tables de données pour créer de nouvelles tables etc...

MySQL : SGBDR de GNU sous licence GPL

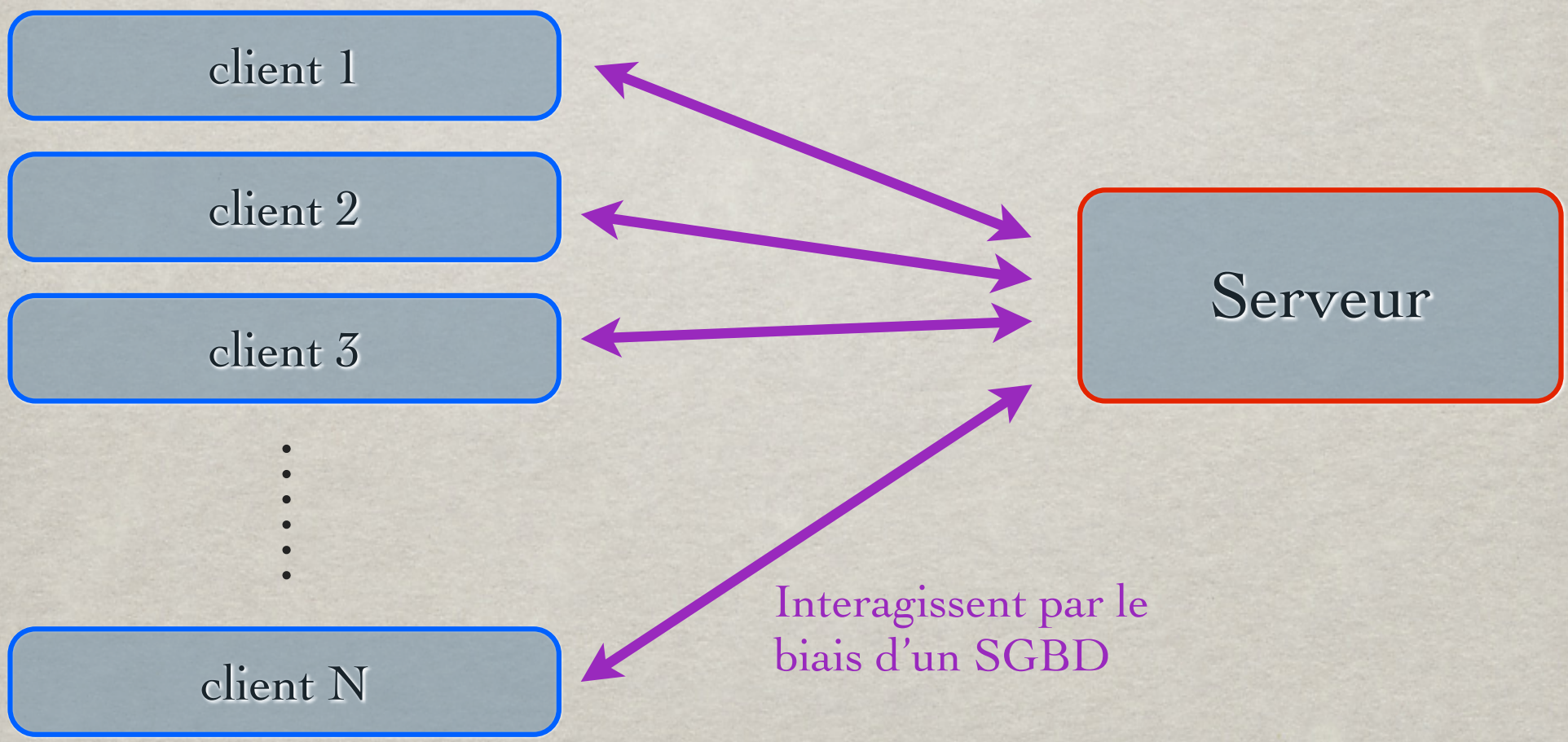
→ Gratuit et utilisé entre autres par Facebook & Google

Autres SGBDR :

- Oracle Database
- Microsoft Access
- Apache Derby
- DB2 IBM
- OpenOffice.org Base
- etc...

Concept de client-serveur & architecture 3/3

A l'origine : logique **client - serveur**



SGBD : Système de gestion de base de données

Système de gestion de base de données : SGBD

Une base de données est un ensemble structuré de données informatiques stockées dans des fichiers (généralement compressés) sur un serveur (ordinateurs distants)

Le SGBD : Système de Gestion de Base de Données.

- C'est un intermédiaire entre l'utilisateur et les fichiers physiques.

- Il donne une représentation intuitive des données sous forme de tables

Objectifs du SGBD :

- Faciliter la compréhension, la manipulation des données.
- Réduire les temps d'accès aux données.
- permettre l'ajout de contraintes :
  - respecter le format des données.
  - donner des droits (confidentialité des données).

Exemple : Un client veut acheter en ligne un article unique. Pendant qu'il le consulte, l'article n'apparaît pas pour les autres utilisateurs car il est réservé au premier.

→ Système de verrou !

Architecture trois tiers :

Couche données :

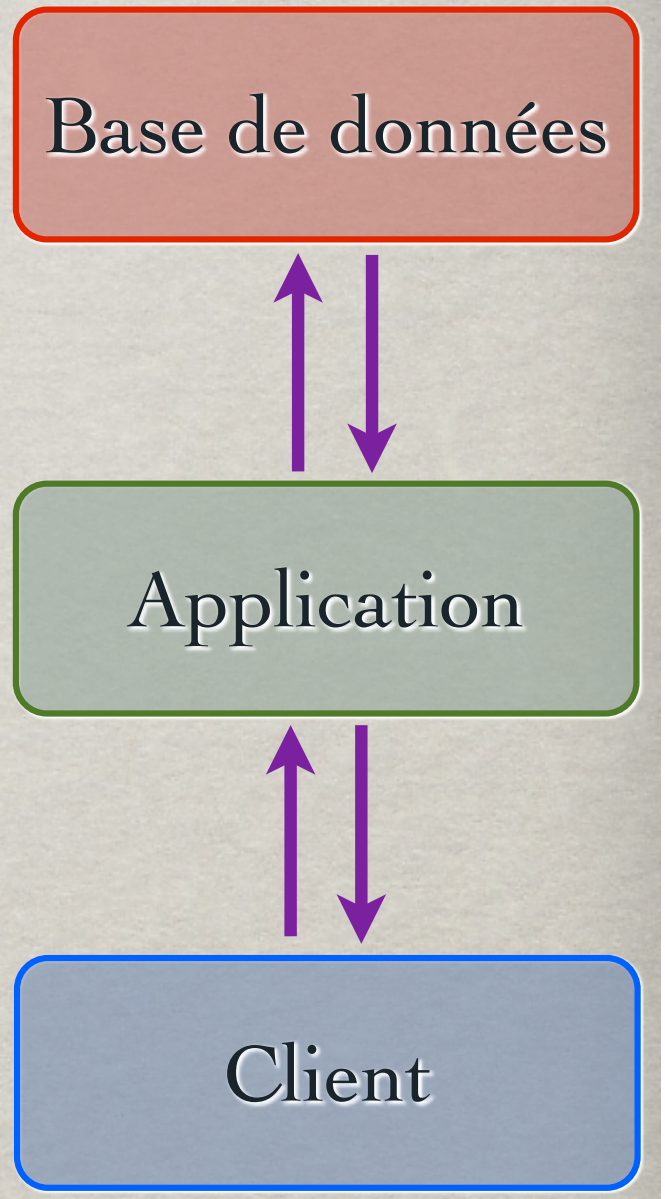
- Gère/conserve/protège les données.
- donne accès aux données physiques en adaptant la requête reçue, à la nature du support.

Application :

- Programme qui génère une requête adaptée
- exécute la requête vers la base de donnée.
- renvoie le résultat à l'utilisateur

Interaction utilisateur :

- Interroge la base de données.
- reçoit le résultat.
  (interface graphique, site web, etc..)



Modèle en trois couches

Base de données physique

Conservation des données :

C'est un point crucial, les données ne peuvent être perdues [banques, assurances, santé, etc ..]
- Les données sont régulièrement sauvegardées [mémoires, disques durs, bandes magnétiques]
- Les serveurs peuvent s'arrêter en cas de panne, il y a donc des serveurs miroirs.

Organisation & transfert des données :

La difficulté réside dans la diminution du temps pour trouver les données.
Différentes stratégies peuvent être employées pour stocker les données :

- **Structure arborescente :** (disque dure PC)
- **Organisation séquentielle :**
  - chaque nouvelle donnée est mise à la suite des données déjà enregistrées (facile à stocker).
  - PB : il faut balayer toute la base de donnée pour trouver les données [**coût linéaire**]

- **Organisation triée :** recherche par dichotomie [**coût logarithmique**]

  Une organisation triée des données est très avantageuse pour trouver où lire la donnée, ou encore pour savoir où écrire la donnée. Elle nécessite toutefois d'avoir indexé les données.
  On réalise pour cela une «**clé primaire**», c-à-d un identifiant strictement croissant
  [ex : ordre alphabétique].

  PB : tout ajout de donnée nécessite de ré-écrire l'ensemble de la base selon l'ordre de classement.

II Représentation tabulaire des bases de données relationnelles

- Exemple concret
- Vocabulaire des bases de données
- Schéma relationnel

Exemple simple et concret d'une base de donnée

La ménagerie :

Nous allons prendre en exemple une base de données formées de trois tables.

Il s'agit ici de décrire une ménagérie contenant des chiens de chats,
des perroquets et des tortues !

- La première table renseigne tous les animaux et leurs particularités.
- La seconde renseigne toutes les races présentes et leurs particularités.
- La troisième renseigne toutes les espèces présentes et leurs particularités.

Une Table est une suite de colonnes (un n-uplet !), appelées «attributs» qui sont renseignées ligne par ligne :

- A chaque attribut correspond un type de donnée [entier, caractère, date, etc....]
- La table peut ainsi être exportée dans un fichier CSV.

Table : Animal

phpMyAdmin

id	sexe	date_naissance	nom	commentaires	espece_id	race_id	mere_id	pere_id
1	M	2010-04-05 13:43:00	Rox	Mordille beaucoup	1	1	18	22
2	*NULL*	2010-03-24 02:23:00	Roucky	*NULL*	2	*NULL*	40	30
3	F	2010-09-13 15:02:00	Schtroumpfette	*NULL*	2	4	41	31
4	F	2009-08-03 05:12:00	*NULL*	*NULL*	3	*NULL*	*NULL*	*NULL*
5	*NULL*	2010-10-03 16:44:00	Choupi	Né sans oreille gauche	2	*NULL*	*NULL*	*NULL*
6	F	2009-06-13 08:17:00	Bobosse	Carapace bizarre	3	*NULL*	*NULL*	*NULL*
7	F	2008-12-06 05:18:00	Caroline	*NULL*	1	2	*NULL*	*NULL*
8	M	2008-09-11 15:38:00	Bagherra	*NULL*	2	5	*NULL*	*NULL*
9	*NULL*	2010-08-23 05:18:00	*NULL*	*NULL*	3	*NULL*	*NULL*	*NULL*
10	M	2010-07-21 15:41:00	Bobo	*NULL*	1	*NULL*	7	21
⋮			⋮					⋮
55	M	2008-03-15 18:45:00	Relou	Surpoids	3	*NULL*	*NULL*	*NULL*
56	M	2009-05-25 18:54:00	Bulbizard	*NULL*	3	*NULL*	*NULL*	*NULL*
57	M	2007-03-04 19:36:00	Safran	*NULL*	4	*NULL*	*NULL*	*NULL*
58	M	2008-02-20 02:50:00	Gingko	*NULL*	4	*NULL*	*NULL*	*NULL*
59	M	2009-03-26 08:28:00	Bavard	*NULL*	4	*NULL*	*NULL*	*NULL*
60	F	2009-03-26 07:55:00	Parlotte	*NULL*	4	*NULL*	*NULL*	*NULL*

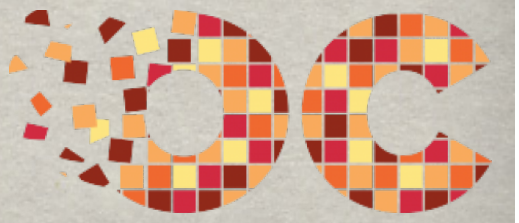
phpMyAdmin

Table : Race

id	nom	espece_id	description
1	Berger allemand	1	Chien sportif et élégant au pelage dense, noir-mar...
2	Berger blanc suisse	1	Petit chien au corps compact, avec des pattes cour...
3	Boxer	1	Chien de taille moyenne, au poil ras de couleur fa...
4	Bleu russe	2	Chat aux yeux verts et à la robe épaisse et argent...
5	Maine coon	2	Chat de grande taille, à poils mi-longs.
6	Singapura	2	Chat de petite taille aux grands yeux en amandes.
7	Sphynx	2	Chat sans poils.

Table : Espece

id	nom_courant	nom_latin	description
1	Chien	Canis canis	Bestiole à quatre pattes qui aime les caresses et ...
2	Chat	Felis silvestris	Bestiole à quatre pattes qui saute très haut et gr...
3	Tortue d'Hermann	Testudo hermanni	Bestiole avec une carapace très dure
4	Perroquet amazone	Alipiopsitta xanthops	Joli oiseau parleur vert et jaune



Définitions ensemblistes de base :

Attribut :

C'est le terme désignant une colonne pour une table donnée [Table.attribut]
A chaque attribut correspond un type de donnée bien précis (entier, réel, NULL, chaîne, etc...)
Soit A_i le i-ème attribut de la table : $A_i \in D_i$ domaine de définition de l'attribut.

L'ensemble des attributs d'une table est donné par le n-uplet : $(A_1, \cdots, A_n)$

Ligne :

Chaque ligne L_i un choix de valeurs pour l'ensemble des attributs de la table.
L'ensemble des lignes possibles est donc le produit cartésien des domaines de tous les attributs :

$$L_i \in D_1 \times D_2 \times \cdots \times D_n = \prod_i D_i$$

Relation : $R \subset \prod_i D_i$

C'est une collection de lignes possibles. La relation est donc représentée par le corps de la table.

Table :

- C'est la donnée de l'ensemble des attributs : entête
- Ainsi que d'un ensemble de lignes représentant la relation : corps de table

Base de données :

Une base de donnée est un ensemble de tables et de correspondances entre les données de ces différentes tables.

clef :

C'est un sous ensemble d'attributs $\{A_{i1}, \cdots, A_{ik}\}$ permettant d'identifier une ligne dans une table ou de faire des liens entre des lignes de différentes tables.

clef primaire :

Elle permet d'identifier de manière unique chaque ligne de la table => «contrainte d'unicité».
Il n'y en a qu'une seule par table. [Rq : elle peut être établie sur plusieurs colonnes.]
L'écriture physique des données est organisée en fonction de la clef primaire.

clef étrangère :

Elle permet de faire un lien avec les données d'une autre table. Les ensembles d'attributs mis en commun entre ces deux tables ont donc le même domaine de définition.

Attributs Type

Animal

	#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra
☐	1	**id**	smallint(5)		UNSIGNED	Non	*Aucune*	AUTO_INCREMENT
☐	2	**sexe**	char(1)	latin1_swedish_ci		Oui	*NULL*	
☐	3	**date_naissance**	datetime			Non	*Aucune*	
☐	4	**nom**	varchar(30)	latin1_swedish_ci		Oui	*NULL*	
☐	5	**commentaires**	text	latin1_swedish_ci		Oui	*NULL*	
☐	6	**espece_id**	smallint(5)		UNSIGNED	Non	*Aucune*	
☐	7	**race_id**	smallint(5)		UNSIGNED	Oui	*NULL*	
☐	8	**mere_id**	smallint(5)		UNSIGNED	Oui	*NULL*	
☐	9	**pere_id**	smallint(5)		UNSIGNED	Oui	*NULL*	

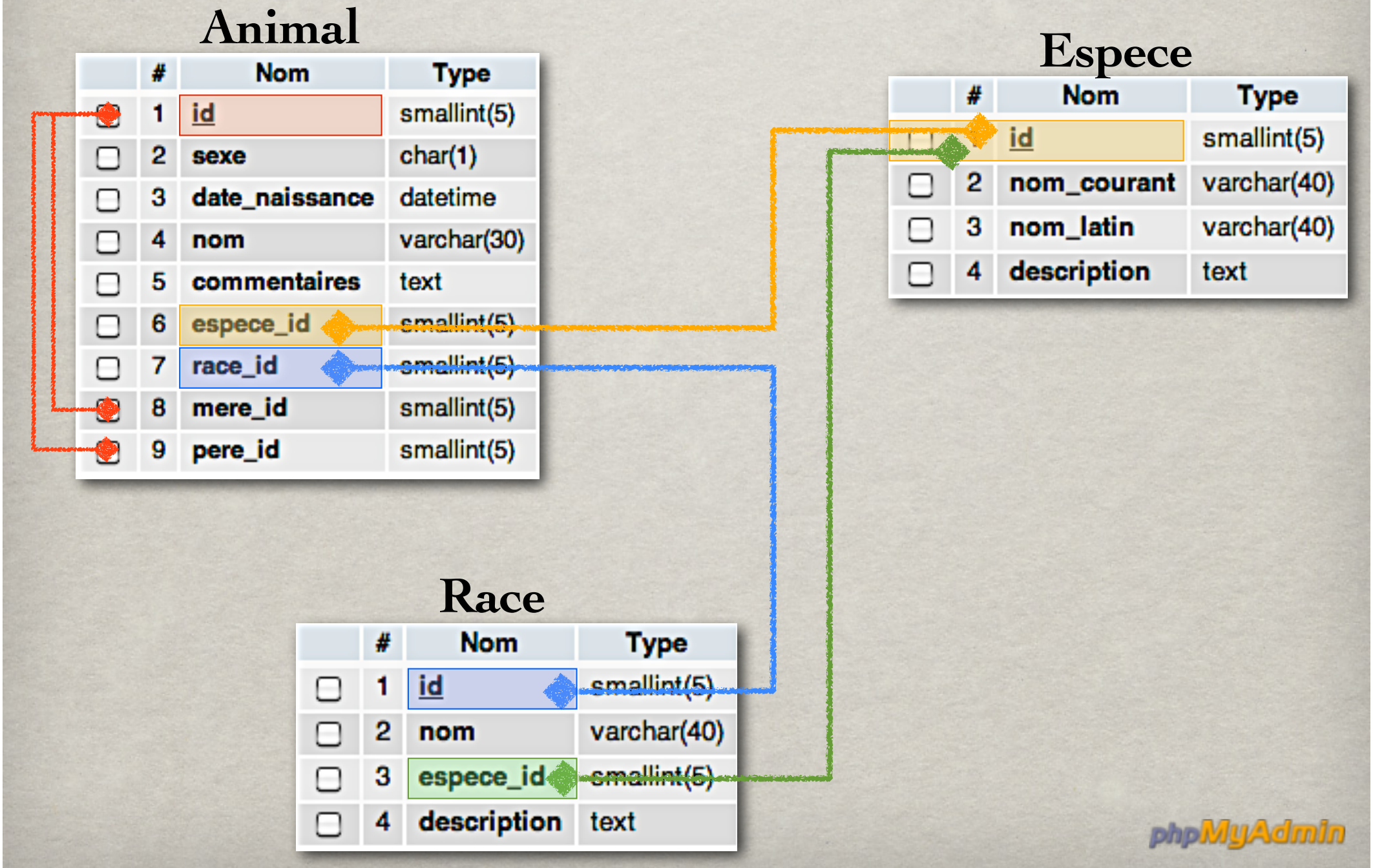
Race

	#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra
☐	1	**id**	smallint(5)		UNSIGNED	Non	*Aucune*	AUTO_INCREMENT
☐	2	**nom**	varchar(40)	latin1_swedish_ci		Non	*Aucune*	
☐	3	**espece_id**	smallint(5)		UNSIGNED	Non	*Aucune*	
☐	4	**description**	text	latin1_swedish_ci		Oui	*NULL*	

Espece

	#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra
☐	1	**id**	smallint(5)		UNSIGNED	Non	*Aucune*	AUTO_INCREMENT
☐	2	**nom_courant**	varchar(40)	latin1_swedish_ci		Non	*Aucune*	
☐	3	**nom_latin**	varchar(40)	latin1_swedish_ci		Non	*Aucune*	
☐	4	**description**	text	latin1_swedish_ci		Oui	*NULL*	

Schéma relationnel de la ménagerie



III Utilisation de MySQL

Opérations de base d'un SGBDR

Objectifs :

- Utilisation de phpMyAdmin pour administrer une base de données.
- Créer/Supprimer des tables, avec leurs attributs.
- Remplir les tables, en gérant les différents types de données.
- Réaliser des opérations de sélection simples.

Création d'une base de données et de ses tables :

```
CREATE DATABASE Base2donnée;

CREATE TABLE Client (id SMALLINT, nom VARCHAR(30), PRIMARY KEY (id) );

    ALTER TABLE Client ADD date DATE NOT NULL ;
  ALTER TABLE Client ADD Prénom VARCHAR(30) NOT NULL AFTER nom ;
  ALTER TABLE Client DROP date ;


CREATE TABLE Commande (id SMALLINT(5), Client_id SMALLINT(5), PRIMARY KEY(id));

  ALTER TABLE Commande ADD plat SMALLINT( 5 ) NOT NULL AFTER Client_id ;
  ALTER TABLE Commande ADD Satisfait CHAR( 1 ) NOT NULL ;


CREATE TABLE Plat(id SMALLINT( 5 ), nom VARCHAR( 30 ), prix REAL);
```

- Toutes ces opérations peuvent être faites à la main (Click-Click....), sans connaître la syntaxe SQL.
- Une fois les tables prêtes, on peut importer les données à partir d'un fichier CSV.
- **La phase essentielle est de bien définir ses tables dès le départ sur papier.**

Remplissage & modification des tables :

```
INSERT INTO Base2donnée.Plat (id, nom, prix) VALUES (NULL, Steak_frites, 12.50);

INSERT INTO Base2donnée.Client (id, nom, date)
VALUES (NULL, Riri, CURRENT_TIME()),
       (NULL, Fifi, CURRENT_TIME()),
       (NULL, Loulou, CURRENT_TIME()),
       (NULL, Donald, CURRENT_TIME()),
       (NULL, Mickey, CURRENT_TIME()),
       (NULL, Minie, CURRENT_TIME());

UPDATE Base2donnée.Client SET Prénom = Duck WHERE Client.id =1;
UPDATE Base2donnée.Client SET Prénom = Mouse WHERE Client.id =5;


INSERT INTO Base2donnée.Plat (id, nom, prix)
VALUES (NULL, Hamburger, 9.50),
       (NULL, Salade, 5.75),
       (NULL, Soda, 5.50),
       (NULL, Spaghetti, 10.50),
       (NULL, Double Burger, 15.0);

INSERT INTO Base2donnée.Commande (id ,Client_id ,plat ,Satisfait)
VALUES (NULL , 1, 3, Y);
```

```
ALTER TABLE Client CHANGE nom prénom VARCHAR( 30 ) CHARACTER SET latin1
COLLATE latin1_swedish_ci NULL DEFAULT NULL ;


INSERT INTO Base2donnée.Commande (id , Client_id , Date , plat, Satisfait)
VALUES ( NULL , 1, 2014-03-20 09:30:28, 3, 'N'),
        ( NULL , 2, 2014-03-27 11:00:00, 6, 'Y'),
            ( NULL , 3, 2014-04-06 10:13:09, 3, 'Y'),
            ( NULL , 4, 2014-04-01 12:00:00, 2, 'Y'),
            ( NULL , 4, 2014-03-24 12:20:00, 5, 'N'),
            ( NULL , 5, 2014-04-08 13:00:00, 1, 'Y'),
            ( NULL , 6, 2014-04-03 12:00:00, 6, NULL),
            ( NULL , 3, 2014-04-16 15:00:00, 5, 'Y'),
            ( NULL , 1, 2014-04-07 13:00:00, 2, 'N'),
etc.....
            (NULL , 6, 2014-04-07 16:00:00, 5, 'N');
```

Les données peuvent être :

- l'ensemble vide NULL
- des chaînes de caractères, du texte etc ...
- des nombres entiers, réels [de tailles diverses et variées]
- des dates [Il existe une multitude de formats]

Chaque ligne (n-uplet) est unique et indexée par son identifiant (clef primaire) :
=> pas de redondance pour éviter la confusion.

IV Algèbre Relationnelle

IV Algèbre Relationnelle

On appelle algèbre relationnnelle, un ensemble de Relations (ou tables) que l'on munie d'opérations internes, c-à-d des opérations qui agissent sur les éléments de cet ensemble de Relations vers ce même ensemble.

Opérations de l'algèbre relationnelle :

Opérations ensemblistes (binaires) :
=> Union, Intersection, Différence, Produit Cartésien.

Opérations spécifiquement relationnelle :
=> Unaire : Selection, Projection.
=> Binaire : Jointure.

Fonctions d'agrégation :
=> Min, Max, Moyenne, comptage, statistiques etc ....

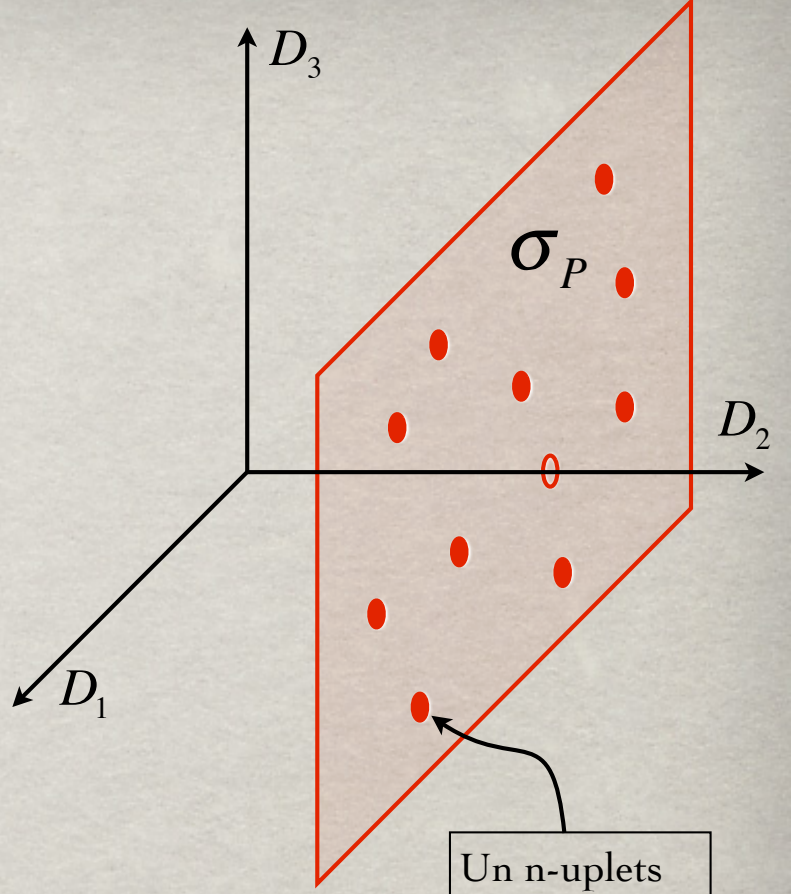
Concrètement ces opérations permettent :

- de combiner des relations (créer de nouvelles tables) pour mettre à jour des liens
- de filtrer les données de ces tables [rechercher une information]
- de réaliser des opérations de comptage [réaliser une statistiques, isolé un élément particulier etc...]

Sélection :

On sélectionne les n-uplets qui vérifient la proposition P :

$$\sigma_P \begin{cases} \prod_i D_i \to \prod_i D_i \\ R \mapsto \sigma_P(R) \end{cases}$$



Exemple :

A	B	C
a	b	1
d	a	2
c	b	3
a	b	4
e	e	5

R

$\sigma_{B='b'}$

A	B	C
a	b	1
c	b	3
a	b	4

$\sigma_P(R)$

Exemple concret : «Trouver tous les chiens et les chats mâles»

$$\sigma_{sexe='M' \ AND \ (espece\_id=1 \ OR \ espece\_id=2)}(Animal)$$

SQL

```
SELECT * FROM Animal
WHERE  (
     sexe = 'M' AND (espece_id = 1 OR espece_id = 2)
   );
```

id	sexe	date_naissance	nom	commentaires	espece_id	race_id	mere_id	pere_id
23	M	2009-03-05 13:54:00	Zoulou	NULL	1	3	NULL	NULL
24	M	2007-04-12 05:23:00	Cartouche	NULL	1	NULL	NULL	NULL
25	M	2006-05-14 15:50:00	Zambo	NULL	1	1	NULL	NULL
26	M	2006-05-14 15:48:00	Samba	NULL	1	1	NULL	NULL
27	M	2008-03-10 13:40:00	Moka	NULL	1	3	NULL	NULL
28	M	2006-05-14 15:40:00	Pilou	NULL	1	1	NULL	NULL
8	M	2008-09-11 15:38:00	Bagherra	NULL	2	5	NULL	NULL
29	M	2009-05-14 06:30:00	Fiero	NULL	2	6	NULL	NULL
30	M	2007-03-12 12:05:00	Zonko	NULL	2	5	NULL	NULL
31	M	2008-02-20 15:45:00	Filou	NULL	2	4	NULL	NULL

«Trouver tous les animaux nés entre avril 2008 et octobre 2008»

$$\sigma_{01/04/08<date\_naissance<31/10/08}(Animal)$$

SQL

```
SELECT id, nom, race_id, sexe, date_naissance FROM Animal
WHERE (date_naissance > '2008-04-01' AND date_naissance < '2008-10-31')
```

id	nom	race_id	sexe	date_naissance
42	Bilba	5	F	2008-04-20 03:20:00
34	Capou	5	M	2008-04-20 03:22:00
39	Zara	5	F	2008-04-20 03:26:00
8	Bagherra	5	M	2008-09-11 15:38:00

On se limite ici aux attributs : (id, nom, race_id, sexe, date_naissance) dans cet ordre

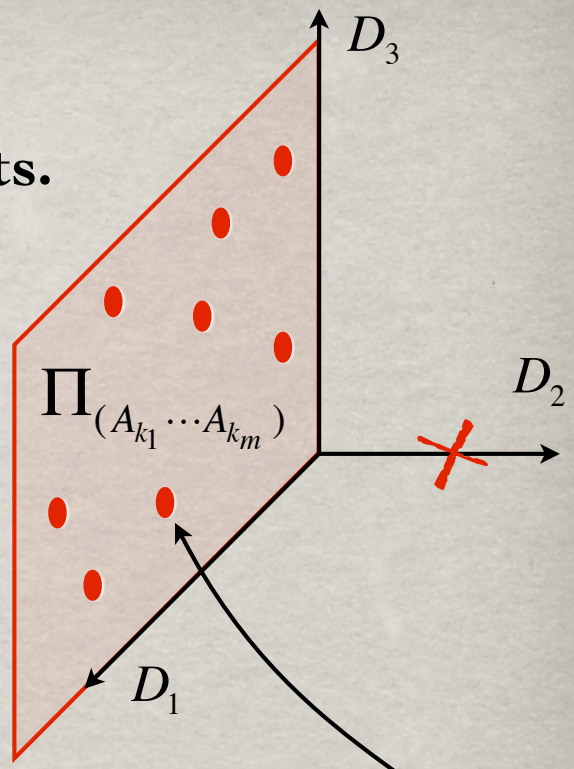
SELECT * --> donnerait tous les attributs

Projection :

On réduit la relation à un sous-ensemble de ses attributs.

On ne garde que certaines colonnes de la table : les n-uplets deviennent des m-uplets avec m < n :

$$\Pi_{(A_{k_1},A_{k_2},\cdots,A_{k_m})} \begin{cases} \prod_i D_i \to \prod_{j\in[k_1\cdots k_m]} D_j \\ R \mapsto \Pi_{(A_{k_1},A_{k_2},\cdots,A_{k_m})}(R) \end{cases}$$



Exemple :

A	B	C
a	b	1
d	a	2
c	b	3
a	b	4
e	e	5

R

$\Pi_{(A,C)}$

A	C
a	1
d	2
c	3
a	4
e	5

$\Pi_{(A,C)}(R)$

L'information sur D2, soit B est écartée.

Exemple concret :

« Lister le nom, la date de naissance et l'identifiant de chaque animal »

$$\Pi_{(nom,\ date\_naissance,\ id)}(Animal)$$

SQL

```
SELECT nom, date_naissance, id
FROM Animal;
```

nom	date_naissance	id
Rox	2010-04-05 13:43:00	1
Roucky	2010-03-24 02:23:00	2
Schtroumpfette	2010-09-13 15:02:00	3
NULL	2009-08-03 05:12:00	4
Choupi	2010-10-03 16:44:00	5
Bobosse	2009-06-13 08:17:00	6
Caroline	2008-12-06 05:18:00	7
Bagherra	2008-09-11 15:38:00	8
NULL	2010-08-23 05:18:00	9
Bobo	2010-07-21 15:41:00	10
Canaille	2008-02-20 15:45:00	11
Cali	2009-05-26 08:54:00	12
Rouquine	2007-04-24 12:54:00	13
Fila	2009-05-26 08:56:00	14

Renommage :

Il s'agit simplement de changer le nom d'un Attribut !

$$\rho_{(A_{k_1} \to A'_{k_1}, \cdots, A_{k_p} \to A'_{k_p})} \begin{cases} \prod_i D_i \to \prod_i D_i \\ R \mapsto \rho_{(A_{k_1} \to A'_{k_1}, \cdots, A_{k_p} \to A'_{k_p})}(R) \end{cases}$$

A	B	C
a	b	1
d	a	2
c	b	3
a	b	4
e	e	5

R

$\rho_{(A \to A, B \to E, C \to G)}$

A	E	G
a	b	1
d	a	2
c	b	3
a	b	4
e	e	5

$\rho_{(A \to A, B \to E, C \to G)}(R)$

Youu-piii

```
ALTER TABLE `Commande` CHANGE `Satisfait` `satisfait`
CHAR( 1 ) CHARACTER SET latin1 NULL DEFAULT NULL ;
```

Remplace l'attribut «Satisfait» par «satisfait» : sans majuscule !

En toute rigueur on devrait toujours indiquer le nom d'un attribut en commençant par le nom de table :

`Animal.id` `Animal.race_id` `Race.id`

Car «id» n'a pas la même signification dans différentes tables.

On peut raccourcir les noms des tables avec des ALIAS

```
SELECT R.id, R.nom FROM Race AS R
```

SQL

id	nom
1	Berger allemand
2	Berger blanc suisse
3	Boxer
4	Bleu russe
5	Maine coon
6	Singapura
7	Sphynx

Les ALIAS deviennent vites indispensables :

- Lorsque les requêtes deviennent trop longues
- Lorsque différentes tables ont des attributs de même nom.
- Lorsque l'on utilise plusieurs fois la même table dans une même requête.

Opérations ensemblistes (binaire)

On veut réaliser les opérations de base de la théorie des ensembles entre deux relations. Soient R et S deux relations.

Produit cartésien :

$$R \times S$$

Soient R et S deux relations : $R \subset \prod_{i=1}^{n} D_{A_i}$ $S \subset \prod_{j=1}^{m} D_{A_j'}$

Les domaines Di et Dj pouvant avoir des Attributs communs ou pas du tout.

On appelle produit cartésien de deux relations, l'opération définie sur chacun des espaces produits des deux relations et à valeur dans l'espace produit cartésien de ceux-ci :

qui pour chaque n-uplet de R, lui associe chacun des m-uplets de S pour former un (n+m)-uplets de l'espace produit final.

Soit formellement :

$$R \times S \begin{cases} \left(\prod_{i=1}^{n} D_{A_j}, \prod_{j=1}^{m} D_{A_j'} \right) \rightarrow \prod_{i=1}^{n} D_{A_j} \times \prod_{j=1}^{m} D_{A_j'} \\ \left(R_{(A_1 \cdots A_n)}, S_{(A_1' \cdots, A_m')} \right) \mapsto R_{(A_1 \cdots A_n)} \times S_{(A_1' \cdots, A_m')} = T_{(A_1 \cdots A_n, A_1' \cdots, A_m')} \end{cases}$$

Plus concrètement :

A	B
1	1
1	2
3	4

R

$\times$

C	D	E
a	b	a
a	b	c
b	a	a

S

$\times$ →

$R \times S$

A	B	C	D	E
1	1	a	b	a
1	1	a	b	c
1	1	b	a	a
1	2	a	b	a
1	2	a	b	c
1	2	b	a	a
3	4	a	b	a
3	4	a	b	c
3	4	b	a	a

RQ : Dans certains cas il peut y avoir des doublons
- interdit en théorie des ensembles.
- tolérés par SQL.

Ex : *Race* × *Espece*

On obtient au final N.M éléments dans la table produit

```
SELECT * FROM Race, Espece;
```

SQL

id	nom	espece_id	description	id	nom_courant	nom_latin	description
1	Berger allemand	1	Chien sportif et élégant au pelage dense, noir-mar...	1	Chien	Canis canis	Bestiole à quatre pattes qui aime les caresses et ...
1	Berger allemand	1	Chien sportif et élégant au pelage dense, noir-mar...	2	Chat	Felis silvestris	Bestiole à quatre pattes qui saute très haut et gr...
1	Berger allemand	1	Chien sportif et élégant au pelage dense, noir-mar...	3	Tortue d'Hermann	Testudo hermanni	Bestiole avec une carapace très dure
1	Berger allemand	1	Chien sportif et élégant au pelage dense, noir-mar...	4	Perroquet amazone	Alipiopsitta xanthops	Joli oiseau parleur vert et jaune
2	Berger blanc suisse	1	Petit chien au corps compact, avec des pattes cour...	1	Chien	Canis canis	Bestiole à quatre pattes qui aime les caresses et ...
2	Berger blanc suisse	1	Petit chien au corps compact, avec des pattes cour...	2	Chat	Felis silvestris	Bestiole à quatre pattes qui saute très haut et gr...
2	Berger blanc suisse	1	Petit chien au corps compact, avec des pattes cour...	3	Tortue d'Hermann	Testudo hermanni	Bestiole avec une carapace très dure
2	Berger blanc suisse	1	Petit chien au corps compact, avec des pattes cour...	4	Perroquet amazone	Alipiopsitta xanthops	Joli oiseau parleur vert et jaune
3	Boxer	1	Chien de taille moyenne, au poil ras de couleur fa...	1	Chien	Canis canis	Bestiole à quatre pattes qui aime les caresses et ...
3	Boxer	1	Chien de taille moyenne, au poil ras de couleur fa...	2	Chat	Felis silvestris	Bestiole à quatre pattes qui saute très haut et gr...
3	Boxer	1	Chien de taille moyenne, au poil ras de couleur fa...	3	Tortue d'Hermann	Testudo hermanni	Bestiole avec une carapace très dure

Etc .....

Etc .....

Attention : on distinguera bien les deux «id» : Race.id et Espece.id

Ex : *Espece* × *Espece*

Rq : pour utiliser deux fois la même table on utilise un «ALIAS» AS

```
SELECT * FROM Espece AS E1, Espece AS E2;
```

SQL

id	nom_courant	nom_latin	description	id	nom_courant	nom_latin	description
1	Chien	Canis canis	Bestiole à quatre pattes qui aime les caresses et ...	1	Chien	Canis canis	Bestiole à quatre pattes qui aime les caresses et ...
2	Chat	Felis silvestris	Bestiole à quatre pattes qui saute très haut et gr...	1	Chien	Canis canis	Bestiole à quatre pattes qui aime les caresses et ...
3	Tortue d'Hermann	Testudo hermanni	Bestiole avec une carapace très dure	1	Chien	Canis canis	Bestiole à quatre pattes qui aime les caresses et ...
4	Perroquet amazone	Alipiopsitta xanthops	Joli oiseau parleur vert et jaune	1	Chien	Canis canis	Bestiole à quatre pattes qui aime les caresses et ...
1	Chien	Canis canis	Bestiole à quatre pattes qui aime les caresses et ...	2	Chat	Felis silvestris	Bestiole à quatre pattes qui saute très haut et gr...
2	Chat	Felis silvestris	Bestiole à quatre pattes qui saute très haut et gr...	2	Chat	Felis silvestris	Bestiole à quatre pattes qui saute très haut et gr...
3	Tortue d'Hermann	Testudo hermanni	Bestiole avec une carapace très dure	2	Chat	Felis silvestris	Bestiole à quatre pattes qui saute très haut et gr...
4	Perroquet amazone	Alipiopsitta xanthops	Joli oiseau parleur vert et jaune	2	Chat	Felis silvestris	Bestiole à quatre pattes qui saute très haut et gr...
1	Chien	Canis canis	Bestiole à quatre pattes qui aime les caresses et ...	3	Tortue d'Hermann	Testudo hermanni	Bestiole avec une carapace très dure
2	Chat	Felis silvestris	Bestiole à quatre pattes qui saute très haut et gr...	3	Tortue d'Hermann	Testudo hermanni	Bestiole avec une carapace très dure
3	Tortue d'Hermann	Testudo hermanni	Bestiole avec une carapace très dure	3	Tortue d'Hermann	Testudo hermanni	Bestiole avec une carapace très dure
4	Perroquet amazone	Alipiopsitta xanthops	Joli oiseau parleur vert et jaune	3	Tortue d'Hermann	Testudo hermanni	Bestiole avec une carapace très dure
1	Chien	Canis canis	Bestiole à quatre pattes qui aime les caresses et ...	4	Perroquet amazone	Alipiopsitta xanthops	Joli oiseau parleur vert et jaune
2	Chat	Felis silvestris	Bestiole à quatre pattes qui saute très haut et gr...	4	Perroquet amazone	Alipiopsitta xanthops	Joli oiseau parleur vert et jaune
3	Tortue d'Hermann	Testudo hermanni	Bestiole avec une carapace très dure	4	Perroquet amazone	Alipiopsitta xanthops	Joli oiseau parleur vert et jaune
4	Perroquet amazone	Alipiopsitta xanthops	Joli oiseau parleur vert et jaune	4	Perroquet amazone	Alipiopsitta xanthops	Joli oiseau parleur vert et jaune

Les doublons ne sont pas éliminés par SQL a priori : E1.chat ≠ E2.chat

Jointure :

On souhaite trouver tous les chiens de race «Berger allemand» :

Lorsque la quantité de données est importante, on ne sait pas quel numéro de Animal.race_id est celui d'une race en particulier.

En revanche une sélection dans la table race au nom «Berger allemand» nous donnerait tout de suite le renseignement que l'on pourrait alors utiliser pour trouver les animaux en question dans la table Animal.

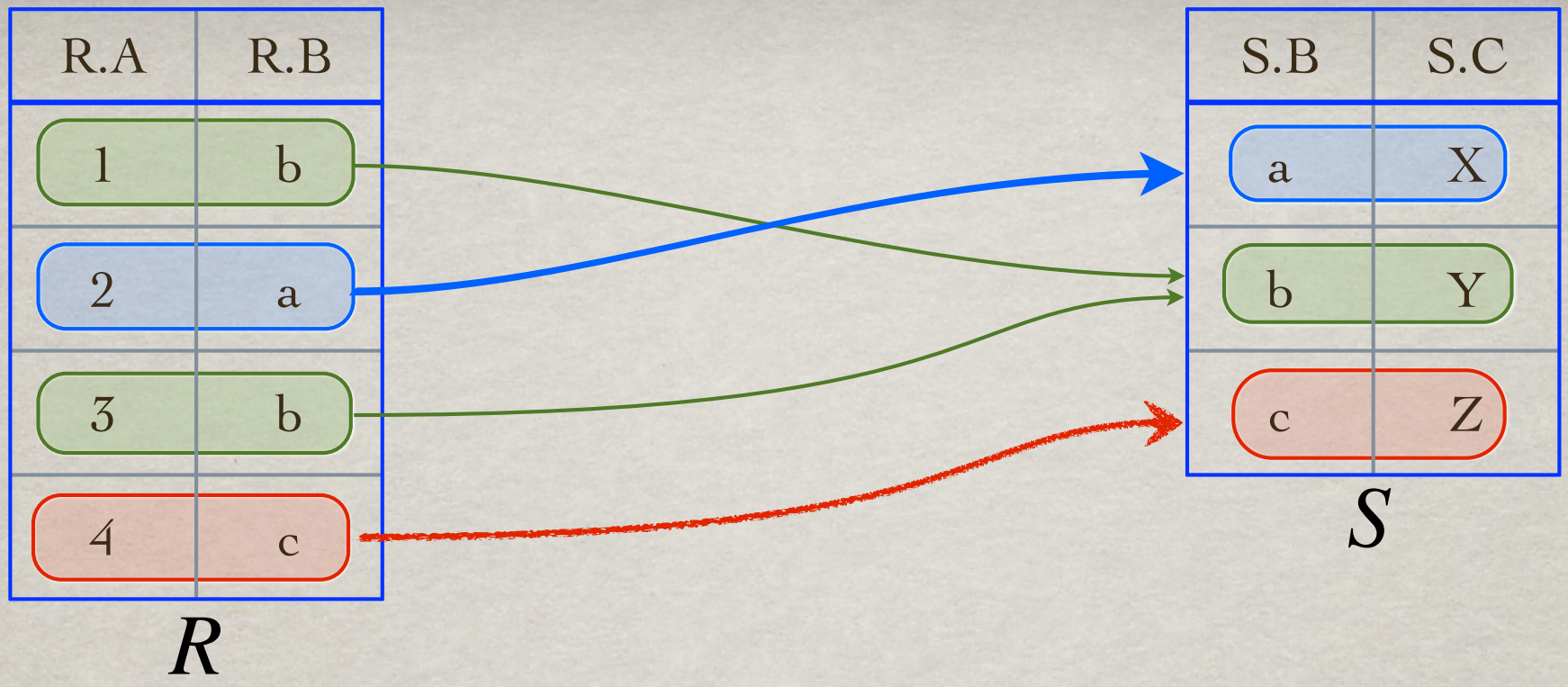
L'algèbre relationnelle permet de faire cette recherche en combinant deux opérations : produit cartésien puis sélection.

Exemple :

Soient deux relations ayant un attribut en commun : $R_{(A,B)}$ et $S_{(B,C)}$

On veut lister tous les attributs (A, B, C) pour lesquels l'attribut B est commun aux deux tables :

ex : **A -> nom de l'animal** **B -> race de l'animal** **C -> description de la race**



$$R \underset{R.B=S.B}{\triangleright\!\triangleleft} S$$

A	B	C
1	b	Y
2	a	X
3	b	Y
4	c	Z

PB : Quel algorithme peut décrire cette mise en relation ?

Mise en relation sans intérêt

$R \times S$

R.A	R.B	S.B	S.C
1	b	a	X
1	b	b	Y
1	b	c	Z
2	a	a	X
2	a	b	Y
2	a	c	Z
3	b	a	X
3	b	b	Y
3	b	c	Z
4	c	a	X
4	c	b	Y
4	c	c	Z

TB

A	B
1	b
2	a
3	b
4	c

R

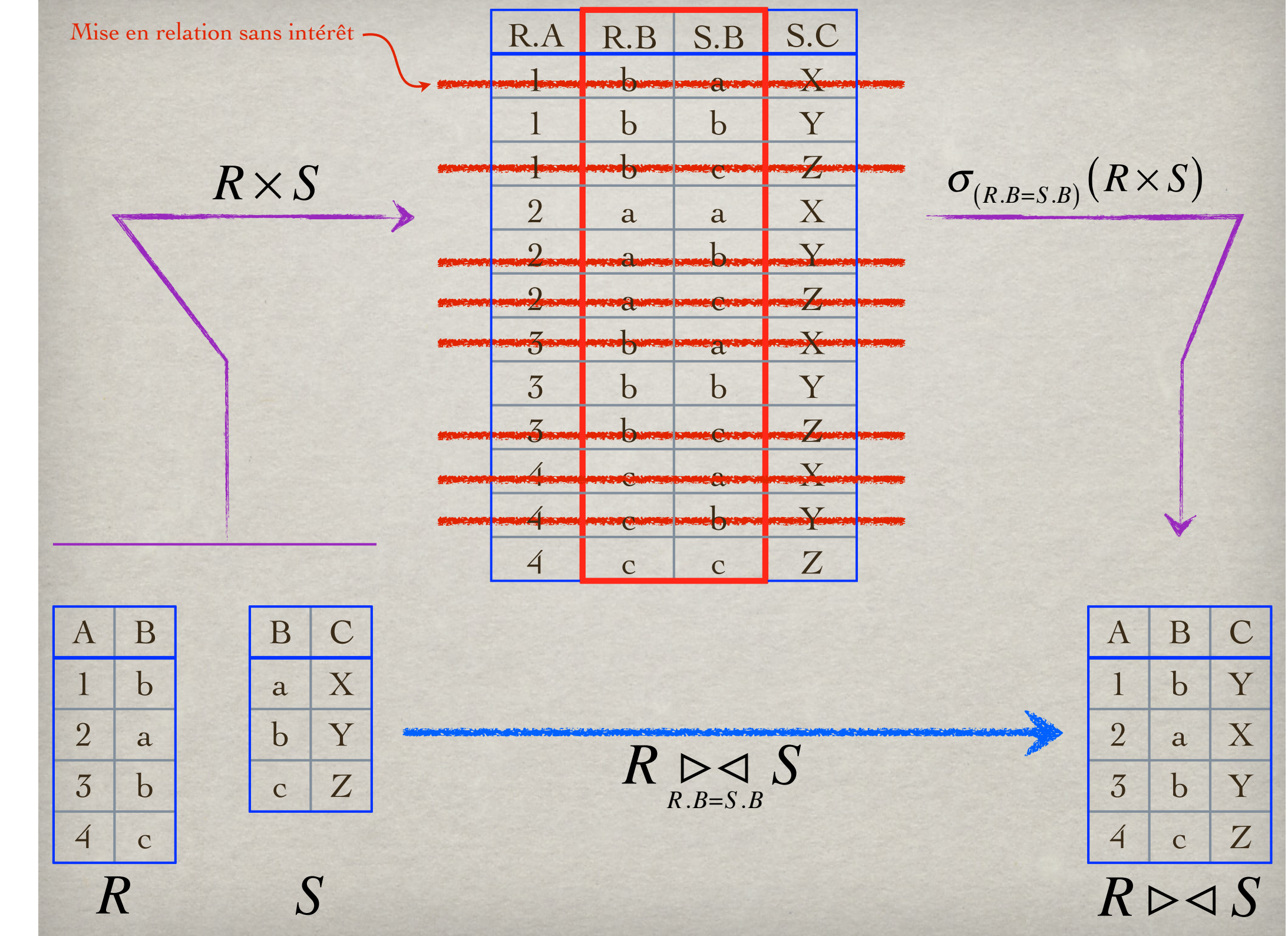
B	C
a	X
b	Y
c	Z

S

$R \triangleright\!\triangleleft_{R.B=S.B} S$

A	B	C
1	b	Y
2	a	X
3	b	Y
4	c	Z

$R \triangleright\!\triangleleft S$


Mise en relation sans intérêt
$R \times S$
$\sigma_{(R.B=S.B)}(R \times S)$
R.A R.B S.B S.C
1 b a X
1 b b Y
1 b c Z
2 a a X
2 a b Y
2 a c Z
3 b a X
3 b b Y
3 b c Z
4 c a X
4 c b Y
4 c c Z
A B
1 b
2 a
3 b
4 c
R
B C
a X
b Y
c Z
S
$R \rhd\lhd_{R.B=S.B} S$
A B C
1 b Y
2 a X
3 b Y
4 c Z
$R \rhd\lhd S$

On voit toutefois que la jointure a sélectionné plusieurs races B possibles.
Il nous suffit alors de sélectionner la race «Berger allemand dans la liste obtenue»

$$\sigma_{(R.B="Berger\ \ allemand")}\left(\sigma_{(R.B=S.B)}\left(R\times S\right)\right)$$

SQL

```
SELECT * FROM Animal INNER JOIN Race ON Animal.race_id = Race.id
WHERE Race.nom = "Berger allemand";
```

id	sexe	date_naissance	nom	commentaires	espece_id	race_id	mere_id	pere_id	id	nom	espece_id	description
1	M	2010-04-05 13:43:00	Rox	Mordille beaucoup	1	1	18	22	1	Berger allemand	1	Chien sportif et élégant au pelage dense, noir-mar...
13	F	2007-04-24 12:54:00	Rouquine	*NULL*	1	1	*NULL*	*NULL*	1	Berger allemand	1	Chien sportif et élégant au pelage dense, noir-mar...
18	F	2007-04-24 12:59:00	Zira	*NULL*	1	1	*NULL*	*NULL*	1	Berger allemand	1	Chien sportif et élégant au pelage dense, noir-mar...
20	M	2007-04-24 12:45:00	Balou	*NULL*	1	1	*NULL*	*NULL*	1	Berger allemand	1	Chien sportif et élégant au pelage dense, noir-mar...
22	M	2007-04-24 12:42:00	Bouli	*NULL*	1	1	*NULL*	*NULL*	1	Berger allemand	1	Chien sportif et élégant au pelage dense, noir-mar...
25	M	2006-05-14 15:50:00	Zambo	*NULL*	1	1	*NULL*	*NULL*	1	Berger allemand	1	Chien sportif et élégant au pelage dense, noir-mar...
26	M	2006-05-14 15:48:00	Samba	*NULL*	1	1	*NULL*	*NULL*	1	Berger allemand	1	Chien sportif et élégant au pelage dense, noir-mar...
28	M	2006-05-14 15:40:00	Pilou	*NULL*	1	1	*NULL*	*NULL*	1	Berger allemand	1	Chien sportif et élégant au pelage dense, noir-mar...

Bilan :

Nous avons 60 animaux et 7 races :

```sql
SELECT * FROM Animal, Race
```

SQL

7 x 60 = 420 animaux
dans le produit cartésien.

```sql
SELECT * FROM Animal INNER JOIN Race
ON Animal.race_id = Race.id
```

SQL

34 animaux dans la jointure
tous de pures races.

```sql
SELECT * FROM Animal INNER JOIN Race
ON Animal.race_id = Race.id
WHERE Race.nom = "Berger allemand"
```

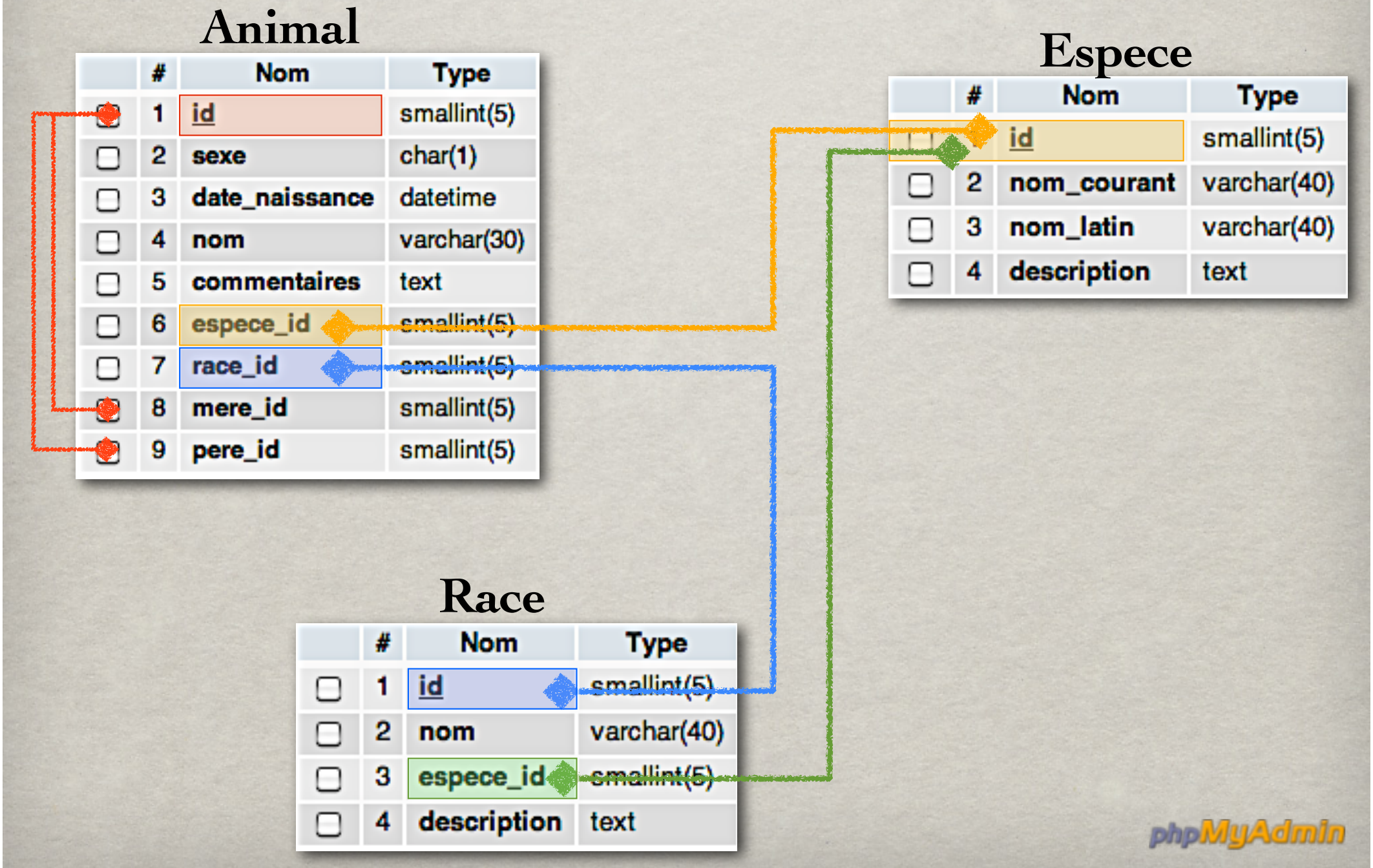
SQL

8 pures races «Berger Allemand»
après sélection.

RQ :

Sans l'attribut commun Animal.race_id=Race.id la jointure est impossible :
On voit à nouveau que la structure des tables est un point essentiel.
Il faut y songer dès la conception de la base de données.

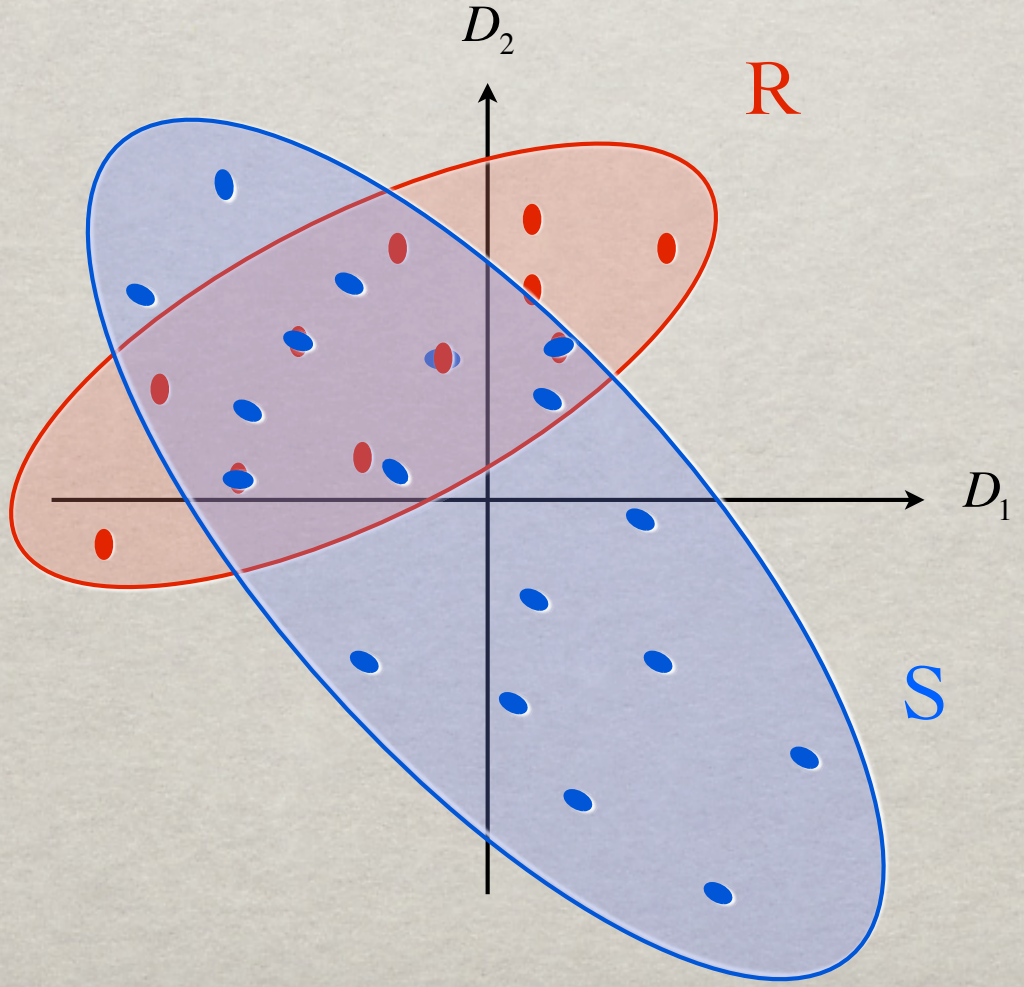
Schéma relationnel de la ménagerie



Union :

Soient deux relations R et S de a et b n-uplets respectivement.
On peut faire la réunion de ces deux collections au sens de la théorie des ensembles, mais pour en faire une nouvelle relation (sous la forme de table de a+b lignes),
il faut que R et S possèdent exactement les mêmes attributs :

Ex : 2D



R

A	B
a	b
a	c
d	e

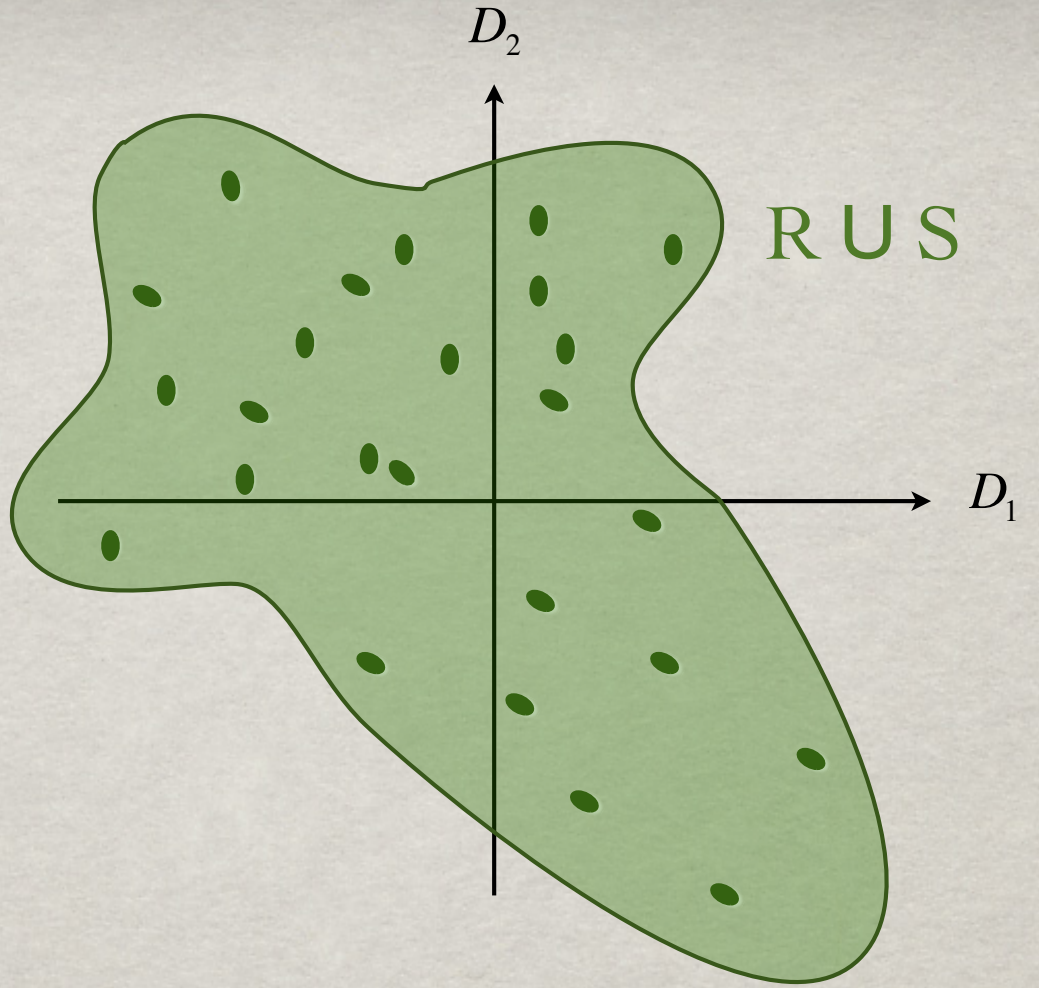
S

A	B
a	b
a	e
d	e
f	g

A	B
a	b
a	c
d	e

U

A	B
a	b
a	e
d	e
f	g



R U S

A	B
a	b
a	c
d	e
a	e
f	g

En théorie des ensembles, il ne peut pas y avoir de doublons,
on a donc supprimé les lignes en doubles !

Quel est l'expression du cardinal de R U S ?

Trouver tous les animaux de races «Berger blanc suisse» ou de race «Maine coon»

```sql
SELECT * FROM Animal INNER JOIN Race ON Race.id = Animal.Race_id
WHERE Race.nom = "Berger blanc suisse"

UNION

SELECT * FROM Animal INNER JOIN Race ON Race.id = Animal.Race_id
WHERE Race.nom = "Maine coon";
```

SQL

12 animaux : 4 bergers blancs suisses (chiens) et 8 Maine coon (chats)

id	sexe	date_naissance	nom	commentaires	espece_id	race_id	mere_id	pere_id	id	nom	espece_id	description
7	F	2008-12-06 05:18:00	Caroline	*NULL*	1	2	*NULL*	*NULL*	2	Berger blanc suisse	1	Petit chien au corps compact, avec des pattes cour...
12	F	2009-05-26 08:54:00	Cali	*NULL*	1	2	*NULL*	*NULL*	2	Berger blanc suisse	1	Petit chien au corps compact, avec des pattes cour...
14	F	2009-05-26 08:56:00	Fila	*NULL*	1	2	*NULL*	*NULL*	2	Berger blanc suisse	1	Petit chien au corps compact, avec des pattes cour...
⋮			⋮			⋮						⋮
40	F	2007-03-12 12:00:00	Milla	*NULL*	2	5	*NULL*	*NULL*	5	Maine coon	2	Chat de grande taille, à poils mi-longs.
42	F	2008-04-20 03:20:00	Bilba	Sourde de l oreille droite a 80%	2	5	*NULL*	*NULL*	5	Maine coon	2	Chat de grande taille, à poils mi-longs.
43	F	2007-03-12 11:54:00	Cracotte	*NULL*	2	5	*NULL*	*NULL*	5	Maine coon	2	Chat de grande taille, à poils mi-longs.

```sql
SELECT * FROM Animal WHERE sexe = 'M'

UNION ALL

SELECT * FROM Animal WHERE espece_id =2

ORDER BY id ASC
```

SQL

On peut autoriser les doublons .... Avec **UNION ALL**

32	M	2007-03-12 12:07:00	Farceur	*NULL*	2	5	*NULL*	*NULL*
32	M	2007-03-12 12:07:00	Farceur	*NULL*	2	5	*NULL*	*NULL*
33	M	2006-05-19 16:17:00	Caribou	*NULL*	2	4	*NULL*	*NULL*
33	M	2006-05-19 16:17:00	Caribou	*NULL*	2	4	*NULL*	*NULL*
34	M	2008-04-20 03:22:00	Capou	*NULL*	2	5	*NULL*	*NULL*
34	M	2008-04-20 03:22:00	Capou	*NULL*	2	5	*NULL*	*NULL*
35	M	2006-05-19 16:56:00	Raccou	Pas de queue depuis la naissance	2	4	*NULL*	*NULL*
35	M	2006-05-19 16:56:00	Raccou	Pas de queue depuis la naissance	2	4	*NULL*	*NULL*
36	M	2009-05-14 06:42:00	Boucan	*NULL*	2	6	*NULL*	*NULL*
36	M	2009-05-14 06:42:00	Boucan	*NULL*	2	6	*NULL*	*NULL*
37	F	2006-05-19 16:06:00	Callune	*NULL*	2	4	*NULL*	*NULL*
38	F	2009-05-14 06:45:00	Boule	*NULL*	2	6	*NULL*	*NULL*
39	F	2008-04-20 03:26:00	Zara	*NULL*	2	5	*NULL*	*NULL*

Différence :

Soient deux relations R et S de a et b n-uplets respectivement.
On veut ici construire la relation des éléments de R qui ne sont pas dans S et inversement.

R

A	B
a	b
a	c
d	e

S

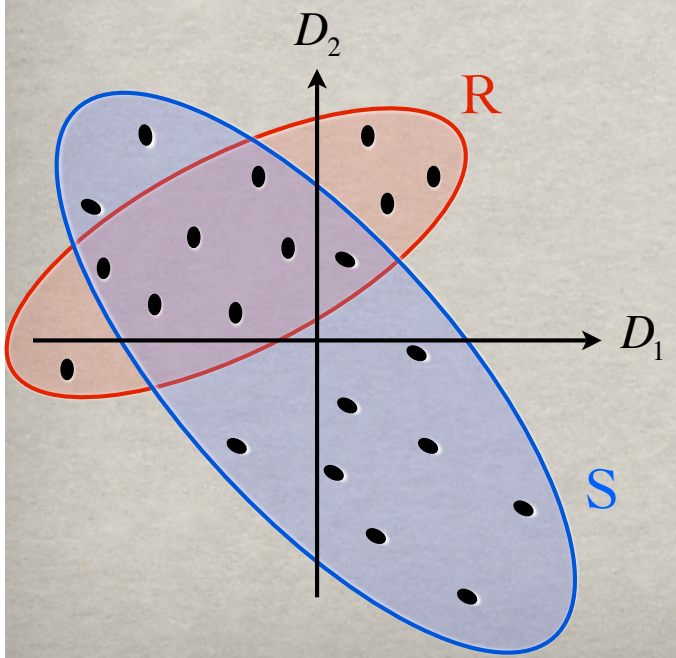
A	B
a	b
a	e
d	e
f	g

R - S

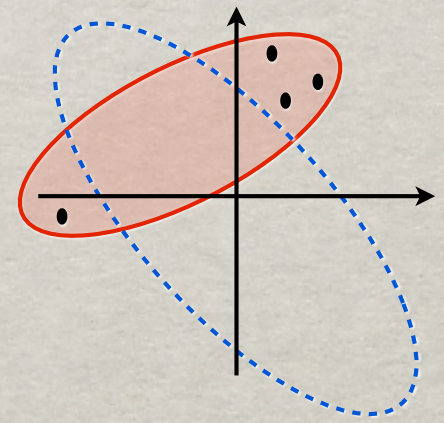
A	B
a	c

S - R

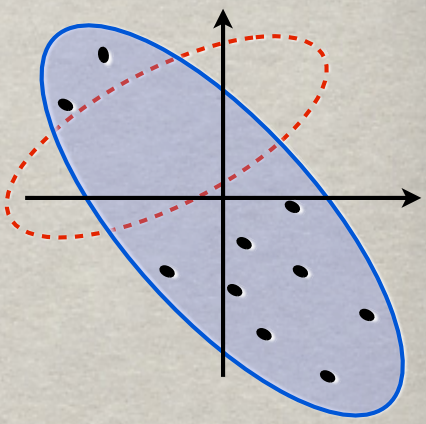
A	B
a	e
f	g



R-S



S-R



R - S ≠ S - R

R : Tous les animaux de l'espèce Chat
S : Tous les animaux de sexe Masculin

NOT IN

SQL

```sql
SELECT Animal.id, Animal.sexe, Animal.nom, Animal.espece_id,
       Espece.id, Espece.nom_courant

FROM Animal INNER JOIN Espece ON Espece.id = Animal.espece_id
  WHERE Especce.nom_courant = 'Chat'
  AND (Animal.id, Animal.sexe) NOT IN (
                              SELECT Animal.id, Animal.sexe FROM Animal
                              WHERE Animal.sexe = 'M'
                                    )
```

Rq : on utilise ici ce que l'on appelle des requêtes imbriquées

R - S

id	sexe	nom	espece_id	id	nom_courant
2	*NULL*	Roucky	2	2	Chat
3	F	Schtroumpfette	2	2	Chat
5	*NULL*	Choupi	2	2	Chat
37	F	Callune	2	2	Chat
38	F	Boule	2	2	Chat
39	F	Zara	2	2	Chat
40	F	Milla	2	2	Chat
41	F	Feta	2	2	Chat
42	F	Bilba	2	2	Chat
43	F	Cracotte	2	2	Chat
44	F	Cawette	2	2	Chat

R - S :

«Tous les animaux de l'espèce Chat, qui ne soient pas masculins.»

```
SELECT Animal.id, Animal.sexe, Animal.nom, Animal.espece_id,
       Espece.id, Espece.nom_courant

FROM Animal INNER JOIN Espece ON Espece.id = Animal.espece_id
WHERE Animal.sexe = 'M'
AND Especce.nom_courant NOT IN (
                        SELECT Espece.nom_courant FROM Espece
                        WHERE Espece.nom_courant = 'Chat'
                                )
```

requêtes imbriquées

Rq : La jointure est ici nécessaire pour accéder au nom_courant

S - R :

«Tous les animaux de sexe masculin,
qui ne soient pas de l'espèce Chats.»

NOT IN

id	sexe	nom	espece_id	id	nom_courant
1	M	Rox	1	1	Chien
10	M	Bobo	1	1	Chien
20	M	Balou	1	1	Chien
21	M	Pataud	1	1	Chien
22	M	Bouli	1	1	Chien
23	M	Zoulou	1	1	Chien
24	M	Cartouche	1	1	Chien
25	M	Zambo	1	1	Chien
26	M	Samba	1	1	Chien
27	M	Moka	1	1	Chien
28	M	Pilou	1	1	Chien
53	M	Spoutnik	3	3	Tortue d'Hermann
54	M	Bubulle	3	3	Tortue d'Hermann
55	M	Relou	3	3	Tortue d'Hermann
56	M	Bulbizard	3	3	Tortue d'Hermann
57	M	Safran	4	4	Perroquet amazone
58	M	Gingko	4	4	Perroquet amazone
59	M	Bavard	4	4	Perroquet amazone

La Différence est une opération non commutative : R - S ≠ S - R

RQ :

Dans les cas particulier de ces exemples, une syntaxe plus simple permet d'obtenir le même résultat en manipulant des expressions logiques.

```
SELECT Animal.id, Animal.sexe, Animal.nom, Animal.espece_id,
       Espece.id, Espece.nom_courant

FROM Animal INNER JOIN Espece ON Espece.id = Animal.espece_id

 WHERE
    Espece.nom_courant = 'Chat'
 AND
     (NOT (Animal.sexe = 'M') OR Animal.sexe IS NULL)
```

On supprime de cette façon le deuxième SELECT,
Mais il n'y a alors plus de notion de Différence entre deux relations.

Intersection :

Soient deux relations R et S de a et b n-uplets respectivement. On veut ici construire la relation des éléments qui sont à la fois dans R et dans S.

R

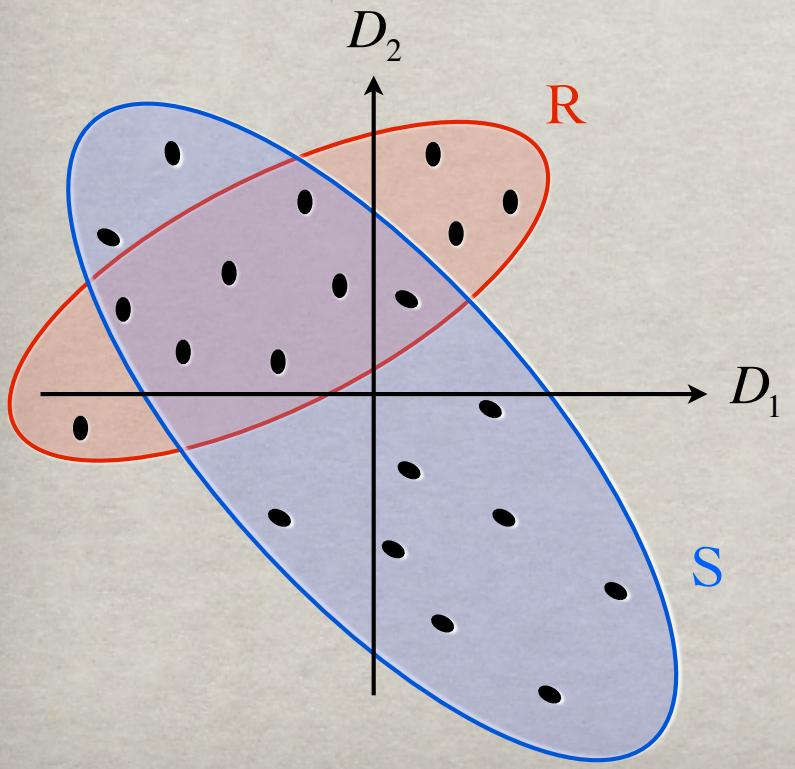
A	B
a	b
a	c
d	e

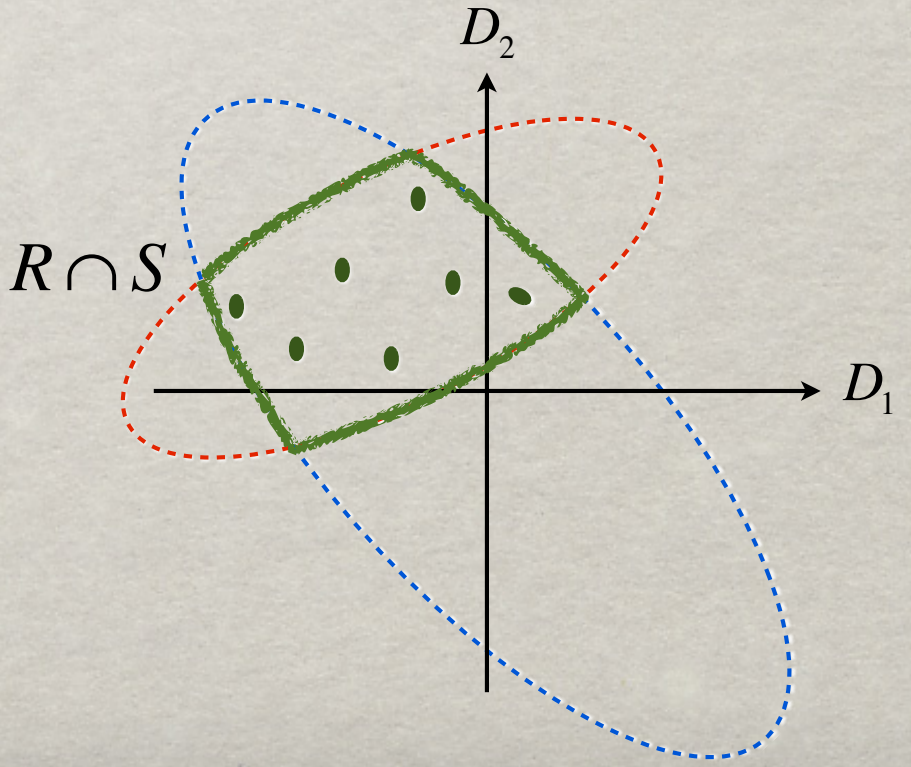
S

A	B
a	b
a	e
d	e
f	g

$R \cap S$

A	B
a	b
d	e





```sql
SELECT Animal.id, Animal.sexe, Animal.nom, Animal.espece_id,
       Espece.id, Espece.nom_courant

FROM Animal INNER JOIN Espece ON Espece.id = Animal.espece_id
WHERE Animal.sexe = 'M'
AND Espece.nom_courant IN (
                           SELECT Espece.nom_courant FROM Espece
                           WHERE Espece.nom_courant = 'Chat'
                          )
```

IN

requêtes imbriquées

```sql
SELECT Animal.id, Animal.sexe, Animal.nom,Animal.espece_id,
       Espece.id, Espece.nom_courant

FROM Animal INNER JOIN Espece ON Espece.id = Animal.espece_id
     WHERE Especе.nom_courant = 'Chat'
     AND (Animal.id, Animal.sexe) IN (
                                      SELECT Animal.id, Animal.sexe FROM Animal
                                      WHERE Animal.sexe = 'M'
                                     )
```

IN

requêtes imbriquées

On remplace simplement ~~NOT IN~~ par IN :

Les deux requêtes donnent le même résultat,
car **l'intersection est un opération commutative !**

$$R \cap S = S \cap R$$

id	sexe	nom	espece_id	id	nom_courant
8	M	Bagherra	2	2	Chat
29	M	Fiero	2	2	Chat
30	M	Zonko	2	2	Chat
31	M	Filou	2	2	Chat
32	M	Farceur	2	2	Chat
33	M	Caribou	2	2	Chat
34	M	Capou	2	2	Chat
35	M	Raccou	2	2	Chat
36	M	Boucan	2	2	Chat

Exercice : Montrer que : $R \cap S = R - (R - S)$

Table : Animal

phpMyAdmin

id	sexe	date_naissance	nom	commentaires	espece_id	race_id	mere_id	pere_id
1	M	2010-04-05 13:43:00	Rox	Mordille beaucoup	1	1	18	22
2	*NULL*	2010-03-24 02:23:00	Roucky	*NULL*	2	*NULL*	40	30
3	F	2010-09-13 15:02:00	Schtroumpfette	*NULL*	2	4	41	31
4	F	2009-08-03 05:12:00	*NULL*	*NULL*	3	*NULL*	*NULL*	*NULL*
5	*NULL*	2010-10-03 16:44:00	Choupi	Né sans oreille gauche	2	*NULL*	*NULL*	*NULL*
6	F	2009-06-13 08:17:00	Bobosse	Carapace bizarre	3	*NULL*	*NULL*	*NULL*
7	F	2008-12-06 05:18:00	Caroline	*NULL*	1	2	*NULL*	*NULL*
8	M	2008-09-11 15:38:00	Bagherra	*NULL*	2	5	*NULL*	*NULL*
9	*NULL*	2010-08-23 05:18:00	*NULL*	*NULL*	3	*NULL*	*NULL*	*NULL*
10	M	2010-07-21 15:41:00	Bobo	*NULL*	1	*NULL*	7	21
⋮			⋮					⋮
55	M	2008-03-15 18:45:00	Relou	Surpoids	3	*NULL*	*NULL*	*NULL*
56	M	2009-05-25 18:54:00	Bulbizard	*NULL*	3	*NULL*	*NULL*	*NULL*
57	M	2007-03-04 19:36:00	Safran	*NULL*	4	*NULL*	*NULL*	*NULL*
58	M	2008-02-20 02:50:00	Gingko	*NULL*	4	*NULL*	*NULL*	*NULL*
59	M	2009-03-26 08:28:00	Bavard	*NULL*	4	*NULL*	*NULL*	*NULL*
60	F	2009-03-26 07:55:00	Parlotte	*NULL*	4	*NULL*	*NULL*	*NULL*

Les fonctions d'agrégation :

COUNT() permet de réaliser différent comptage

Combien d'animaux pure race :

```
SELECT count(race_id) FROM Animal WHERE race_id > 0
```

Combien d'animaux pure race classés par race :

```
SELECT race_id, count(race_id) FROM Animal WHERE race_id > 0
GROUP BY race_id
```

Si on veut afficher le nom explicite de la race :

```
SELECT R.nom, count(A.race_id)
FROM Animal as A INNER JOIN Race as R ON A.race_id = R.id
GROUP BY A.race_id
```

MIN() MAX() AVG() permettent de réaliser des statistiques

Trouver le produit le moins cher :

```
SELECT min(prix) FROM Produit
```

Trouver le prix moyen :

```
SELECT avg(prix) FROM Produit
```

Trouver les articles moins cher que la moyenne :

```
SELECT nom FROM Produit
WHERE prix < avg(prix)
```

Que font les requêtes suivantes :

```
SELECT nom, prix, prix/max(prix)*100, prix/avg(prix) FROM Produit
```

```
SELECT categorie, avg(prix) FROM Produit
GROUP BY categorie
```

Division euclidienne :

Il s'agit «simplement» de montrer qu'une relation peut s'écrire comme un produit cartésien de deux relations avec un reste éventuel en union :

$$R = Q \times S \cup T$$

Exemple :

R

A	B	C	D	E
1	a	1	b	a
1	a	2	b	a
1	a	4	a	a
1	b	1	b	a
1	b	2	b	c
1	b	4	a	a
2	a	1	b	c
2	a	4	a	c
3	c	1	b	b
3	c	2	b	b
3	c	4	a	b
3	c	3	c	b

Division euclidienne :

Il s'agit «simplement» de montrer qu'une relation peut s'écrire comme un produit cartésien de deux relations avec un reste éventuel en union :

$$R = Q \times S \cup T$$

A	B	C	D	E
1	a	1	b	a
1	a	2	b	a
1	a	4	a	a
1	b	1	b	a
1	b	2	b	c
1	b	4	a	a
2	a	1	b	c
2	a	4	a	c
3	c	1	b	b
3	c	2	b	b
3	c	4	a	b
3	c	3	c	b

Division euclidienne :

Il s'agit «simplement» de montrer qu'une relation peut s'écrire comme un produit cartésien de deux relations avec un reste éventuel en union :

$$R = Q \times S \cup T$$

R

A	B	C	D	E
1	a	1	b	a
1	a	2	b	a
1	a	4	a	a
1	b	1	b	a
1	b	2	b	c
1	b	4	a	a
2	a	1	b	c
2	a	4	a	c
3	c	1	b	b
3	c	2	b	b
3	c	4	a	b
3	c	3	c	b

Reste

Reste

S

C	D
1	b
2	b
4	a

Q

A	B	E
1	a	a
3	c	b

Plus formellement :

Soit deux relations R(A1...An, B1,...Bm) et S(B1,...Bm)

On appelle quotient : $Q \equiv R \div S$ ⟶ Q(A1...An)

Par définition :

$$R \div S = \left\{ (a_1 \ldots a_n) \mid \forall (b_1 \ldots b_m) \in S : (a_1 \ldots a_n, b_1 \ldots b_m) \in R \right\}$$

Une fois trouvé le quotient, on construit le reste T par Différence : $T \equiv R - (Q \times S)$

La division est toujours possible, car dans le pire des cas : Q ={∅} et T = R

Une fois déterminer Q et T, on peut vérifier que : $R = Q \times S \cup T$

La division relationnelle peut s'exprimer uniquement en fonction du **produit cartésien**, de la **projection** et de la **différence**.

On donne l'algorithme suivant pour obtenir le quotient et le reste :

Soit $R_1 = \Pi_{(A_1 \cdots A_n)}(R)$ et $R_2 = \Pi_{(A_1 \cdots A_n)}\left(\left(R_1 \times S\right) - R\right)$

On a : $Q = R \div S = R_1 - R_2$ $T \equiv R - \left(Q \times S\right)$

On peut donc écrire directement :

$$Q = \Pi_{(A_1 \cdots A_n)}(R) - \Pi_{(A_1 \cdots A_n)}\left(\left(R_1 \times S\right) - R\right)$$

Application : Testons l'algorithme sur notre exemple

R

A	B	C	D	E
1	a	1	b	a
1	a	2	b	a
1	a	4	a	a
3	c	1	b	b
3	c	2	b	b
3	c	4	a	b
1	b	1	b	a
1	b	2	b	c
1	b	4	a	a
2	a	1	b	c
2	a	4	a	c
3	c	3	c	b

÷

S

C	D
1	b
2	b
4	a

$R_1 = \Pi_{(A,B,E)}(R)$

A	B	E
1	a	a
1	a	a
1	a	a
3	c	b
3	c	b
3	c	b
1	b	a
1	b	c
1	b	a
2	a	c
2	a	c
3	c	b

Suppression des doublons

R_1

A	B	E
1	a	a
3	c	b
1	b	a
1	b	c
2	a	c

$\times$

S

C	D
1	b
2	b
4	a

$=$

$R_1 \times S$

A	B	E	C	D
1	a	a	1	b
1	a	a	2	b
1	a	a	4	a
3	c	b	1	b
3	c	b	2	b
3	c	b	4	a
1	b	a	1	b
1	b	a	2	b
1	b	a	4	a
1	b	c	1	b
1	b	c	2	b
1	b	c	4	a
2	a	c	1	b
2	a	c	2	b
2	a	c	4	a

Ré-agencement

$R_1 \times S$

A	B	C	D	E
1	a	1	b	a
1	a	2	b	a
1	a	4	a	a
3	c	1	b	b
3	c	2	b	b
3	c	4	a	b
1	b	1	b	a
1	b	2	b	a
1	b	4	a	a
1	b	1	b	c
1	b	2	b	c
1	b	4	a	c
2	a	1	b	c
2	a	2	b	c
2	a	4	a	c

$-$

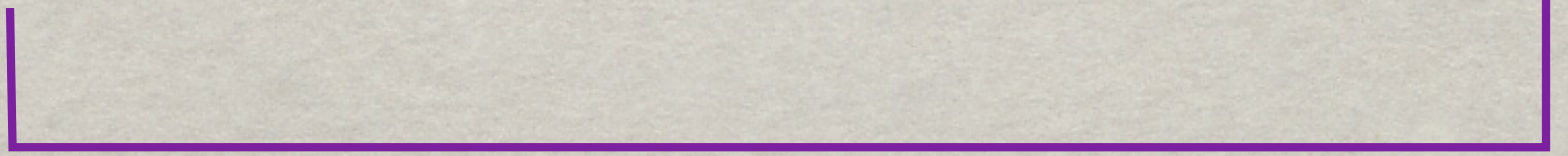
R

A	B	C	D	E
1	a	1	b	a
1	a	2	b	a
1	a	4	a	a
3	c	1	b	b
3	c	2	b	b
3	c	4	a	b
1	b	1	b	a
1	b	2	b	c
1	b	4	a	a
2	a	1	b	c
2	a	4	a	c
3	c	3	c	b

$=$

$R_1 \times S - R$

A	B	C	D	E
1	b	2	b	a
1	b	1	b	c
1	b	4	a	c
2	a	2	b	c



$R_1 \times S - R$

A	B	C	D	E
1	b	2	b	a
1	b	1	b	c
1	b	4	a	c
2	a	2	b	c

$R_2 = \Pi_{(A,B,E)}\left((R_1 \times S) - R\right)$

A	B	E
1	b	a
1	b	c
~~1~~	~~b~~	~~c~~
2	a	c

R_1

A	B	E
1	a	a
3	c	b
~~1~~	~~b~~	~~a~~
~~1~~	~~b~~	~~c~~
~~2~~	~~a~~	~~c~~

$-$

R_2

A	B	E
1	b	a
1	b	c
~~1~~	~~b~~	~~c~~
2	a	c

$=$

$R \div S = R_1 - R_2$

A	B	E
1	a	a
3	c	b

$Q = R \div S$

A	B	E
1	a	a
3	c	b

$\times$

S

C	D
1	b
2	b
4	a

$=$

$Q \times S$

A	B	E	C	D
1	a	a	1	b
1	a	a	2	b
1	a	a	4	a
3	c	b	1	b
3	c	b	2	b
3	c	b	4	a

$=$

$Q \times S$

A	B	C	D	E
1	a	1	b	a
1	a	2	b	a
1	a	4	a	a
3	c	1	b	b
3	c	2	b	b
3	c	4	a	b

R

A	B	C	D	E
~~1~~	~~a~~	~~1~~	~~b~~	~~a~~
~~1~~	~~a~~	~~2~~	~~b~~	~~a~~
~~1~~	~~a~~	~~4~~	~~a~~	~~a~~
~~3~~	~~c~~	~~1~~	~~b~~	~~b~~
~~3~~	~~c~~	~~2~~	~~b~~	~~b~~
~~3~~	~~c~~	~~4~~	~~a~~	~~b~~
1	b	1	b	a
1	b	2	b	c
1	b	4	a	a
2	a	1	b	c
2	a	4	a	c
3	c	3	c	b

$-$

$Q \times S$

A	B	C	D	E
1	a	1	b	a
1	a	2	b	a
1	a	4	a	a
3	c	1	b	b
3	c	2	b	b
3	c	4	a	b

$=$

T

A	B	C	D	E
1	b	1	b	a
1	b	2	b	c
1	b	4	a	a
2	a	1	b	c
2	a	4	a	c
3	c	3	c	b

Q

A	B	E
1	a	a
3	c	b

×

S

C	D
1	b
2	b
4	a

$Q \times S$

A	B	C	D	E
1	a	1	b	a
1	a	2	b	a
1	a	4	a	a
3	c	1	b	b
3	c	2	b	b
3	c	4	a	b

∪

T

A	B	C	D	E
1	b	1	b	a
1	b	2	b	c
1	b	4	a	a
2	a	1	b	c
2	a	4	a	c
3	c	3	c	b

Reste

$R = Q \times S \cup T$

A	B	C	D	E
1	a	1	b	a
1	a	2	b	a
1	a	4	a	a
3	c	1	b	b
3	c	2	b	b
3	c	4	a	b
1	b	1	b	a
1	b	2	b	c
1	b	4	a	a
2	a	1	b	c
2	a	4	a	c
3	c	3	c	b

L'ordre des n-uplets n'est pas pris en compte :

il faudra éventuellement trier

Application concrète :

La division relationnelle permet de formuler des requettes telles que :

«Quels sont les clients qui ont déjà commandés tous les plats ?»

On appelle R la table des commandes (Client_id, plat)et S celle des plats (Plat.id)

Soit Q la réponse :

Q **x** S est la table de ces clients en vis à vis de toutes leurs commandes distinctes

R - Q **x** S est le reste : toutes les commandes distinctes des autres clients

Syntaxe MySQL qui traduit l'algorithme :

Quotient

```
SELECT DISTINCT Client_id FROM Commande AS COM1
WHERE NOT EXISTS (
                    SELECT * FROM Plat
                    WHERE NOT EXISTS (

                                        SELECT * FROM Commande AS COM2
                                        WHERE ((COM1.Client_id=COM2.Client_id)
                                        AND (COM2.plat=Plat.id))
                                        )
                    )
```

Reste

```
SELECT * FROM Commande
WHERE id NOT IN (

      SELECT id FROM Commande AS COM1
      WHERE NOT EXISTS (

                SELECT * FROM Plat
                WHERE NOT EXISTS(

                  SELECT * FROM Commande AS COM2
               WHERE ((COM1.Client_id = COM2.Client_id)
               AND (COM2.plat = Plat.id))
                     )
                         )
                 )
```

`EXISTS` : Vaut TRUE lorsque le résultat de la sélection entre parenthèse n'est pas vide

`NOT EXISTS` : Vaut TRUE lorsque le résultat de la sélection entre parenthèse est vide

`DISTINCT` : Elimine les doublons

R

id	Client_id	Date	plat	satisfait
1	1	2014-02-01 12:00:00	3	Y
2	1	2014-03-20 09:30:28	5	Y
3	2	2014-03-27 11:00:00	6	*NULL*
4	3	2014-04-01 10:13:09	3	Y
5	4	2014-04-03 12:00:00	2	Y
6	4	2014-04-09 12:20:00	5	N
7	5	2014-04-11 13:00:00	1	Y
8	6	2014-04-13 12:00:00	6	Y
9	3	2014-04-16 15:00:00	5	Y
10	1	2014-04-17 13:00:00	2	N
11	6	2014-04-18 16:00:00	5	N
12	1	2014-04-22 08:28:00	4	N
13	3	2014-04-28 10:04:00	4	N
14	2	2014-05-01 11:00:00	4	N
15	6	2014-05-03 11:00:00	4	N
16	5	2014-05-06 00:00:00	4	*NULL*
17	1	2014-05-20 11:40:00	7	N
18	1	2014-05-28 14:11:07	6	Y
19	1	2014-04-29 10:06:00	1	Y
20	3	2014-05-30 10:00:00	1	*NULL*
21	1	2014-06-01 14:00:00	6	*NULL*
22	3	2014-06-04 11:00:00	7	*NULL*
23	3	2014-06-05 00:00:00	2	*NULL*
24	6	2014-06-07 19:00:00	3	*NULL*
25	1	2014-06-08 00:00:00	3	*NULL*
26	3	2014-04-10 00:00:00	6	*NULL*

S

id	nom	prix
1	pizza	10
2	Steak_frites	12.5
3	Hamburger	9.5
4	Salade	5.75
5	Soda	5.5
6	Spaghetti	10.5
7	Double Burger	15

Q

Client_id
1
3

T

id	Client_id	Date	plat	satisfait
3	2	2014-03-27 11:00:00	6	*NULL*
5	4	2014-04-03 12:00:00	2	Y
6	4	2014-04-09 12:20:00	5	N
7	5	2014-04-11 13:00:00	1	Y
8	6	2014-04-13 12:00:00	6	Y
11	6	2014-04-18 16:00:00	5	N
14	2	2014-05-01 11:00:00	4	N
15	6	2014-05-03 11:00:00	4	N
16	5	2014-05-06 00:00:00	4	*NULL*
24	6	2014-06-07 19:00:00	3	*NULL*