BDW1 : Bases de données et Programmation Web

SQL : Partie 1 - de la projection à la jointure

Nicolas Lumineau

nicolas.lumineau@univ-lyon1.fr

Université Claude Bernard Lyon 1

Licence 2^e année - 2016/2017

`http://liris.cnrs.fr/nicolas.lumineau/`

$>$ Licence 2 $>$ BDW1

Enseignements de BDW1

1. [BD & Web] Introduction
2. [**BD**]La syntaxe SQL
   - SQL : de la projection à la jointure
   - SQL : sous-requêtes et regroupements
   - Fonctions diverses et commandes SQL
3. [**BD**]Conception de base de données
   - Schéma Entité/Association
   - Du modèle conceptuel au modèle relationnel
4. [**Web**] Programmation Web
   - Le langage HTML/CSS
   - Le langage PHP
     - Interrogation d'une BD via PHP
5. [**BD**] Optimisation de requête
   - Algèbre Relationnelle
   - Transformation d'arbres algébriques

Qu'est ce que SQL ?

SQL = Structured Query Language

SQL est un langage concret pour interagir avec le modèle relationnel :

- Un langage de manipulation de données
- Un langage de description de données
- Un langage pour administrer la base, gérer les contrôles d'accès

SQL est un **langage déclaratif** qui permet à l'utilisateur de déclarer/caractériser un résultat...

... sans avoir à dire comment le calculer !

Historique de SQL

Origine : IBM en 1974

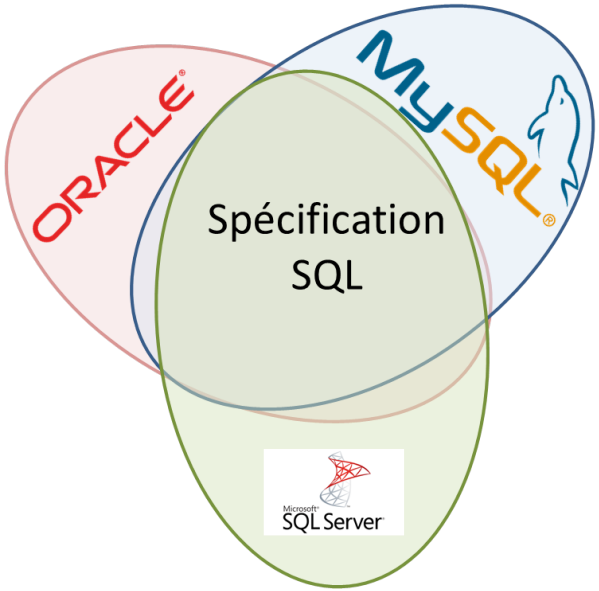
Standards :

- SQL-87 : 1987 (ISO)
- SQL-2 : 1992
- SQL-3 : 1999
- SQL-2003
- SQL-2006
- SQL-2008
- SQL-2011

SQL et les SGBD relationnels

De nombreux SGBD utilisent le langage SQL :

- MySQL
- PostgreSQL
- Oracle
- SQL Server
- MariaDB
- ...



Chaque éditeur de SGBD à sa propre implémentation de la spécification SQL

Il y a la théorie et la pratique

Rappel théorique sur le modèle relationnel

- Il n'y a pas de doublons
- Tous les attributs ont une valeur
- Les tuples ne sont pas ordonnés

et en SQL :

- On a la possibilité d'avoir des doublons ou de les supprimer
- On a la possibilité d'ordonner le résultat des requêtes
- Les attributs peuvent ne pas avoir de valeur
- On n'a pas forcément les raccourcis syntaxiques correspondant à chaque opérateur ensembliste

Partie 3 : Le langage SQL

Au commencement...

L'informaticien dispose de tables implémentées dans un SGBD de type MySQL

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

JoueDans(ida, idf, nom)

idp	idf	nom	sal
1	3	George Valentin	0,3
1	4	Hubert Bonisseur de La Bath	0,2
2	2	Lucy Miller	14
3	1	Bill Cage	20
4	4	Larmina El Akmar Betouche	NULL
4	3	Peppy Miller	NULL
5	5	Antoine Bailleul	1,3
6	1	Rita Vrataski	NULL

Film(idf, titre, annee, genre, real, budget)

idf	titre	annee	genre	real	budget
1	Edge of tomorrow	2014	SF	7	178
2	Lucy	2014	SF	8	40
3	The Artist	2011	Drame	9	9
4	OSS 117: Le Caire, Nid d'espions	2006	NULL	9	14
5	Bienvenue chez les Ch'tis	2008	Comédie	5	11

Pays(code, nom)

code	nom
FR	France
USA	Etats-Unis
UK	Royaume-Uni
AR	Argentine

Dans une requête SQL...

... il y a des clauses SQL et différents mots-clés qui vont permettre de spécifier le résultat que vous souhaitez.

Clauses SQL

SELECT ...
FROM ...
WHERE ...
ORDER BY ...
GROUP BY ...
HAVING ...

Mots-clés SQL

AND , **OR** , **NOT**
DISTINCT , **DESC**
IN , **ANY** , **ALL**
BETWEEN...AND... ...
AND ,
LIKE , **IS** , **NULL**
count, *sum*, *avg*, ...

Apprendre le langage SQL pour interroger une base de données
⇒ Comprendre à quoi correspondent les différentes clause
+ RESPECTER LA SYNTAXE!!!

Une requête SQL

```
SELECT a.nom, a.prenom, count(DISTINCT ja.idf) AS nb
FROM Artiste a JOIN JoueDans ja ON a.ida = ja.ida,
Artiste b JOIN JoueDans jb ON b.ida = jb.ida
WHERE a.nation = b.nation
AND a.idf IN (SELECT f.idf
              FROM Film
              WHERE budget > 15)
AND NOT EXISTS (SELECT *
                FROM joueDans j
                WHERE j.ida <> jb.ida
                AND j.nom LIKE '%Miller%')
GROUP BY a.ida, a.nom, a.prenom
HAVING nb > 1
ORDER BY a.nom, a.prenom DESC;
```

Des clauses et des opérations

Dans une requête SQL de base, il est possible d'exprimer dans :

- une clause **SELECT**
  - une **projection** pour spécifier les attributs qui constitueront le résultat de la requête.
  - un **renommage** pour afficher des attributs avec un nom différent dans le résultat de la requête.
- une clause **WHERE**
  - une **sélection** pour sélectionner les tuples qui seront retournés.
- une clause **ORDER BY**
  - un **ordre** sur l'affichage des tuples retournés.

Dans une requête SQL, il est *toujours* nécessaire d'exprimer l'origine des attributs manipulés dans la requête, *i.e.*, les tables utiles.
$\Rightarrow$ Clause **FROM**

NB : les autres clauses seront vues plus tard car elles impliquent des requêtes SQL plus avancées.

Expression de la Projection

SELECT att_1, att_2, ...
FROM *nomTable* ;

- Dans la clause **SELECT** est spécifiée la liste des attributs qui seront retournés dans le résultat. Chaque nom d'attribut est séparé par une virgule (,).
- Dans la clause **FROM** est spécifiée la table d'où les attributs proviennent.
- La requête se termine par un point-virgule (;)

Remarque importante !

Une requête SQL sans **SELECT** ou sans **FROM** n'est pas correcte.

Exemple 1 de Projection

Donner le nom et le prénom des artistes

SELECT nom, prenom **FROM** *Artiste* ;

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

$\Rightarrow$

Résultat :

prenom	nom
Jean	DUJARDIN
Scarlett	JOHANSSON
Tom	CRUISE
Bérénice	BEJO
Dany	BOON
Emily	BLUNT
Doug	LIMAN
Luc	BESSON
Michel	HAZANAVICIUS

Un raccourci syntaxique pour dire 'tous les attributs' : *

Dans le cas où l'on souhaite spécifier tous les attributs dans la clause **SELECT** , il est possible d'utiliser l'étoile (*).
L'étoile est équivalente à la liste de tous les attributs provenant des tables listées dans la clause **FROM** .

Donner les artistes

SELECT ida, nom, prenom, nation **FROM** *Artiste* ;
$\Longleftrightarrow$ **SELECT** * **FROM** *Artiste* ;

Résultat :

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

Exemple 2 de Projection

Donner la nationalité des artistes

SELECT nation **FROM** *Artiste* ;

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

$\Rightarrow$

Résultat :

nation
FR
USA
USA
FR-AR
FR
UK
USA
FR
FR

Remarque sur l'exemple 2 de Projection

Le résultat précédent n'est pas satisfaisant, car les nationalités présentent dans la base apparaissent autant de fois qu'il y a un/une artiste de cette nationalité.

Pour répondre correctement à cette requête il est nécessaire de supprimer les doublons.

Suppression des doublons : DISTINCT

Le mot clé **DISTINCT** permet de supprimer les tuples en doublon.
Il se place au début de la clause **SELECT** :

SELECT DISTINCT att_1, att_2, ...
FROM *nomTable* ;

Retour à l'exemple :

Donner la nationalité des artistes

SELECT DISTINCT nation **FROM** *Artiste* ;

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

$\Rightarrow$

Résultat :

nation
FR
USA
FR-AR
UK

Renommage d'attribut dans le résultat : AS

Le mot-clé **AS** permet d'associer lors de l'affichage, un nom d'attribut différent du nom utilisé dans la table :

SELECT att_1 **AS** att'_1, att_2 **AS** att'_2, $att_3, \ldots$
FROM *nomTable* ;

- L'attribut att_1 sera renommé en att'_1, et l'attribut att_2 sera renommé en att'_2

Remarque :
Le AS ne permet qu'un renommage de l'attribut uniquement dans l'affichage du résultat. *En aucun cas, la table n'est modifiée !*

Exemple de Renommage

Donner l'identifiant, le nom et la nationalité des artistes. Le nom de l'artiste sera renommé "nomArtiste" et la nationalité sera renommé "pays".

```
SELECT ida, nom AS nomArtiste, nation AS pays
FROM Artiste ;
```

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

$\Rightarrow$

Résultat :

ida	nomArtiste	pays
1	DUJARDIN	FR
2	JOHANSSON	USA
3	CRUISE	USA
4	BEJO	FR-AR
5	BOON	FR
6	BLUNT	UK
7	LIMAN	USA
8	BESSON	FR
9	HAZANAVICIUS	FR

Expression de la sélection

SELECT att_1, att_2, ...
FROM *nomTable*
WHERE *condition* ;

- La clause **WHERE** permet d'exprimer une condition qui caractérisent les tuples qui seront retournés par la requête.
- La condition exprimée dans le **WHERE** est une expression logique qui retourne vrai ou faux. L'expression est appliqué à chaque tuple et seuls sont gardés dans le résultat les tuples rendant l'expression vraie.

Remarque

La clause **WHERE** exprime des contraintes au niveau des tuples, *i.e., des contraintes qui seront testées sur chaque tuple*

La condition dans le **WHERE**

- L'expression exprimant une condition à vérifier peut contenir :
  - une comparaison entre un attribut et une constante
  - une comparaison entre deux attributs
  - une conjonction de comparaisons (comp1 **AND** comp2)
  - une disjonction de comparaisons (comp1 **OR** comp2)
  - des combinaisons de conjonction et/ou de disjonction
  - *des fonctions, des expressions plus complexes* (plus tard)
- Pour les comparaisons, nous avons $=$, $!=$, $<$, $<=$, $>$, $>=$ pour respectivemebnt l'égalité, le différent, le strictement inférieur, l'inférieur, le strictement supérieur et le supérieur.
- Pour les constantes, nous avons différents types de données utilisés pour les constantes : nombres (*1, 1000, 1.5,...*), des chaînes de caractères (*'PIM', 'PAM', 'POUM'...*) et des dates (*'2017-01-23'*) avec un formatage des dates qui peut varier selon le SGBD.

Exemple 1 de sélection

Donner les artistes américain(e)s

```
SELECT *
FROM Artiste
WHERE nation = 'USA';
```

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

$\Rightarrow$

Résultat :

ida	prenom	nom	nation
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
7	Doug	LIMAN	USA

Exemple 2 de sélection

Donner le titre et le budget des films sortis depuis 2010 et ayant un budget strictement supérieur à 12

```
SELECT titre, budget
FROM Film
WHERE annee >= 2010 AND budget > 12;
```

Film(idf, titre, annee, genre, real, budget)

idf	titre	annee	genre	real	budget
1	Edge of tomorrow	2014	SF	7	178
2	Lucy	2014	SF	8	40
3	The Artist	2011	Drame	9	9
4	OSS 117: Le Caire, Nid d'espions	2006	NULL	9	14
5	Bienvenue chez les Ch'tis	2008	Comédie	5	11

$\Rightarrow$

Résultat :

titre	budget
Edge of tomorrow	178
Lucy	40

Exemple 3 de sélection

Donner les film sortis entre 2011 et 2014 ou ceux dont le budget est inférieur à 12

```
SELECT *
FROM Film
WHERE (annee >= 2011 AND annee <= 2014)
 OR budget < 12 ;
```

Film(idf, titre, annee, genre, real, budget)

idf	titre	annee	genre	real	budget
1	Edge of tomorrow	2014	SF	7	178
2	Lucy	2014	SF	8	40
3	The Artist	2011	Drame	9	9
4	OSS 117: Le Caire, Nid d'espions	2006	NULL	9	14
5	Bienvenue chez les Ch'tis	2008	Comédie	5	11

$\Rightarrow$

Résultat :

idf	titre	annee	genre	real	budget
1	Edge of tomorrow	2014	SF	7	178
2	Lucy	2014	SF	8	40
3	The Artist	2011	Drame	9	9
5	Bienvenue chez les Ch'tis	2008	Comédie	5	11

Condition sur un intervalle : BETWEEN...AND...

Dans une clause **WHERE** , il est possible de contraindre un attribut à un intervalle de valeur en utilisant le raccourci syntaxique **BETWEEN...AND...** .

SELECT att_1 , att_2, ...
FROM *nomTable*
WHERE att_1 **BETWEEN** *bmin* **AND** *bmax* ;

- *bmin* représente la valeur minimale pouvant être prise par l'attribut att_1
- *bmax* représente la valeur maximale pouvant être prise par l'attribut att_1

Condition sur un intervalle : BETWEEN...AND... (2)

SELECT att_1 , att_2, ...
FROM *nomTable*
WHERE att_1 **BETWEEN** *bmin* **AND** *bmax* ;

Remarque :

- La notion d'intervalle s'applique non seulement aux nombres, mais aussi aux chaînes de caractères (application de l'ordre lexicographique) et aux dates.
- il est important de noter que le **AND** est associé au **BETWEEN...AND...** . Il ne faut pas le confondre avec un **AND** d'une conjonction de comparaisons.

Retour sur l'exemple 3 de sélection

Donner les film sortis entre 2011 et 2014 ou ceux dont le budget est inférieur à 12

```
SELECT *
FROM Film
WHERE annee BETWEEN 2011 AND 2014
 OR budget < 12 ;
```

Film(idf, titre, annee, genre, real, budget)

idf	titre	annee	genre	real	budget
1	Edge of tomorrow	2014	SF	7	178
2	Lucy	2014	SF	8	40
3	The Artist	2011	Drame	9	9
4	OSS 117: Le Caire, Nid d'espions	2006	NULL	9	14
5	Bienvenue chez les Ch'tis	2008	Comédie	5	11

$\Rightarrow$

Résultat :

idf	titre	annee	genre	real	budget
1	Edge of tomorrow	2014	SF	7	178
2	Lucy	2014	SF	8	40
3	The Artist	2011	Drame	9	9
5	Bienvenue chez les Ch'tis	2008	Comédie	5	11

Condition sur un ensemble de valeurs : **IN**

Dans une clause **WHERE** , il est possible de contraindre un attribut à un ensemble discret de valeurs en utilisant le raccourci syntaxique **IN** .

SELECT att_1 , att_2, ...
FROM *nomTable*
WHERE att_1 **IN** (val_1, val_2,...) ;

- val_i représentent les valeurs possibles pour l'attribut att_1

Exemple 4 de sélection

Donner les artistes de nationalité française ou franco-argentine

```
SELECT *
FROM Artiste
WHERE nation IN ('FR' , 'FR-AR');
```

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

$\Rightarrow$

Résultat :

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

Exemple 4' de sélection

Requête équivalente sans le **IN** :

Donner les artistes de nationalité française ou franco-argentine

```
SELECT *
FROM Artiste
WHERE nation = 'FR' OR nation = 'FR-AR';
```

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

$\Rightarrow$

Résultat :

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

Condition portant sur un motif : **LIKE** ...

Dans une clause **WHERE** , il est possible de contraindre un attribut non pas sur une valeur exacte mais sur un motif.
Un motif est une valeur dont on ne connaît pas exactement tous les caractères pour les chaînes de caractères ou tous les chiffres pour une valeur numérique.

Un motif s'exprime grâce à des caractères joker :

- le _ permet d'exprimer un seul caractère quelconque
- le % permet d'exprimer plusieurs caractères quelconques

Ainsi, pour le motif "P_M", les valeurs 'PIM' et 'PAM' répondent au motif, mais ce n'est pas le cas pour la valeur 'POUM'.
Pour que cela soit le cas, il faudrait considérer le motif 'P%M'.

Condition portant sur un motif : **LIKE** ... (2)

Dans une clause **WHERE** , il est possible de contraindre un attribut sur un motif en utilisant l'opérateur **LIKE** et les caractères joker _ et %.

SELECT att_1 , att_2, ...
FROM *nomTable*
WHERE att_1 **LIKE** "mot_1" ;

- mot_1 représente un motif contenant au moins un caractère joker et qui permettra de filtrer les valeurs possibles pour l'attribut att_1

Exemple 5 de sélection

Donner les artistes de nationalité française

```
SELECT *
FROM Artiste
WHERE nation LIKE "%FR%";
```

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

$\Rightarrow$

Résultat :

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

Exemple 6 de sélection

Donner les artistes dont le nom contient un "O" comme seconde lettre

```
SELECT *
FROM Artiste
WHERE nom LIKE "_O%";
```

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

$\Rightarrow$

Résultat :

ida	prenom	nom	nation
2	Scarlett	JOHANSSON	USA
5	Dany	BOON	FR

Condition portant sur l'absence de valeur : **IS NULL**

Dans une clause **WHERE** , il est possible de contraindre un attribut selon qu'il n'est pas de valeur.

Une valeur non définie est représentée par le mot-clé **NULL** .

Il est possible de filtrer un tuple qui n'a pas de valeur grâce à l'opérateur **IS** couplé au mot-clé **NULL** .

SELECT att_1 , att_2, ...
FROM *nomTable*
WHERE att_1 **IS NULL ;**

- cette requête ne retournera que les tuples pour lesquels l'attribut att_1 n'a pas de valeur.

Exemple 8 de sélection

Donner les films dont on ne connait pas le genre

```
SELECT *
FROM Film
WHERE genre IS NULL;
```

Film(idf, titre, annee, genre, real, budget)

idf	titre	annee	genre	real	budget
1	Edge of tomorrow	2014	SF	7	178
2	Lucy	2014	SF	8	40
3	The Artist	2011	Drame	9	9
4	OSS 117: Le Caire, Nid d'espions	2006	NULL	9	14
5	Bienvenue chez les Ch'tis	2008	Comédie	5	11

Résultat :

$\Rightarrow$

idf	titre	annee	genre	real	budget
4	OSS 117 : Le Caire, Nid d'espions	2006	NULL	9	14

Condition portant sur l'existence de valeurs : IS NOT NULL

A l'inverse, il est possible, dans une clause **WHERE** , de contraindre un attribut pour s'assurer qu'il a bien une valeur. Il suffit dans ce cas d'ajouter **NOT** , entre l'opérateur **IS** et le mot-clé **NULL** .

SELECT att_1 , att_2, ...
FROM *nomTable*
WHERE att_1 **IS NOT NULL ;**

- cette requête ne retournera tous les tuples pour lesquels l'attribut att_1 a une valeur.

Exemple 8 de sélection

Donner les films dont on ne connaît pas le genre

```
SELECT *
FROM Film
WHERE genre IS NOT NULL;
```

Film(idf, titre, annee, genre, real, budget)

idf	titre	annee	genre	real	budget
1	Edge of tomorrow	2014	SF	7	178
2	Lucy	2014	SF	8	40
3	The Artist	2011	Drame	9	9
4	OSS 117: Le Caire, Nid d'espions	2006	NULL	9	14
5	Bienvenue chez les Ch'tis	2008	Comédie	5	11

$\Rightarrow$

Résultat :

idf	titre	annee	genre	real	budget
1	Edge of tomorrow	2014	SF	7	178
2	Lucy	2014	SF	8	40
3	The Artist	2011	Drame	9	9
5	Bienvenue chez les Ch'tis	2008	Comédie	5	11

Premier bilan sur la clause **WHERE**

Nous avons vu, que la clause **WHERE** permet d'exprimer des conditions qui permettent de filtrer des tuples, en fonction :

- d'une valeur constante
- d'une combinaison (conjonction/disjonction) de valeurs constantes
- d'un ensemble de valeurs bornées ou définies
- d'un motif
- de l'absence ou non de valeur

DISTINCT , NULL ...

Nous avons vu précédemment qu'il était possible :

- de permettre ou non la présence de doublons dans le résultat d'une requête
  - **DISTINCT** dans la clause **SELECT**
- de permettre à un attribut de ne pas avoir de valeur
  - symbole **NULL**
  - avec possibilité de filtre grâce à l'opérateur **IS** (éventuellement combiné avec la négation **NOT**)

Intéressons nous maintenant à la possibilité de trier le résultat d'une requête.

Expression du tri : ORDER BY ...

SELECT att_1, att_2, ...
FROM *nomTable*
WHERE *condition*
ORDER BY att_1, att_2, ... ;

- La clause **ORDER BY** permet d'exprimer l'ordre d'apparition des tuples dans le résultat de la requête.
- La clause **ORDER BY** se place toujours en fin de requête.
- La clause **ORDER BY** est suivie d'une liste d'attributs séparés par une virgule (,). Le résultat de la requête est trié par défaut selon l'ordre naturel croissant du premier attribut de la liste (att_1). En cas d'égalité des valeurs sur le premier attribut, le deuxième attribut (att_2) est considéré afin de trier les tuples concernés selon l'ordre naturel croissant de ce dernier. Et ainsi de suite avec les autres attributs de la liste.

Expression du tri : ORDER BY ... (2)

SELECT att_1, att_2, ...
FROM *nomTable*
WHERE *condition*
ORDER BY att_1, att_2 DESC, ... ;

- Dans la clause **ORDER BY** , il est possible de faire suivre le nom d'un attribut par *ASC* ou *DESC* pour indiquer un ordre croissant ou décroissant.
- Il est à noter que l'ordre par défaut est l'ordre croissant. Par conséquence, le mot-clé *ASC* sera très peu utilisé.
- Dans l'exemple, avec l'ajout de *DESC* après l'attribut att_2, le résultat de la requête est trié selon l'ordre naturel croissant de l'attribut att_1 et pour les valeurs commune de att_1, les tuples sont triés selon l'ordre naturel décroissant de l'attribut (att_2).

Expression du tri : ORDER BY ... (3)

La relation d'ordre s'applique naturellement sur les valeurs numériques mais aussi sur les chaînes de caractères et les dates.

Ordre pour les chaînes de caractères

- L'ordre naturel croissant pour les chaînes de caractères est l'ordre lexicographique (ou ordre alphabétique) : A, B, C,...
- L'ordre naturel décroissant est l'ordre lexicographique inverse : Z, Y, X, ...

Ordre pour les dates

- L'ordre naturel croissant pour les dates est l'ordre l'ordre chronologique : 23-01-2017, 27-01-2017,...
- L'ordre naturel décroissant est l'ordre antéchronologique : 27-01-2017, 23-01-2017,...

Exemple 1 de tri

Donner les films de budget supérieur strictement à 10. Le résultat sera trié par année croissante puis par budget décroissant.

```
SELECT *
FROM Film
WHERE budget > 10
ORDER BY annee, budget DESC;
```

Film(idf, titre, annee, genre, real, budget)

idf	titre	annee	genre	real	budget
1	Edge of tomorrow	2014	SF	7	178
2	Lucy	2014	SF	8	40
3	The Artist	2011	Drame	9	9
4	OSS 117: Le Caire, Nid d'espions	2006	NULL	9	14
5	Bienvenue chez les Ch'tis	2008	Comédie	5	11

$\Rightarrow$

Résultat :

idf	titre	annee	genre	real	budget
4	OSS 117 : Le Caire, Nid d'espions	2006	NULL	9	14
5	Bienvenue chez les Ch'tis	2008	Comédie	5	11
1	Edge of tomorrow	2014	SF	7	178
2	Lucy	2014	SF	8	40

Exemple 2 de tri

Donner le nom et la nationalité des artistes dont le prénom contient 3 ou 4 lettres. Le résultat sera trié par nationalité croissante puis par nom croissant.

```
SELECT nom, nation
FROM Artiste
WHERE prenom LIKE "___" OR prenom LIKE "____"
ORDER BY nation, nom ;
```

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

$\Rightarrow$

Résultat :

nom	nation
BESSON	FR
BOON	FR
DUJARDIN	FR
CRUISE	USA
LIMAN	USA

Bilan sur la clause **ORDER BY**

Nous avons vu, que la clause **ORDER BY** permet de spécifier les attributs sur lesquels s'appliqueront un tri pour afficher le résultat. Il est possible d'exprimer

- un ordre sur une valeur numérique, une chaîne de caractères et une date
- que le tri sera croissant (par défaut) ou décroissant (en spécifiant DESC après l'attribut)

Une requête SQL

Les mot-clés que nous avons déjà vu !

```
SELECT a.nom, a.prenom, count(DISTINCT ja.idf) AS nb
FROM Artiste a JOIN JoueDans ja ON a.ida = ja.ida,
Artiste b JOIN JoueDans jb ON b.ida = jb.ida
WHERE a.nation = b.nation
AND a.idf IN (SELECT f.idf
              FROM Film
              WHERE budget > 15)
AND NOT EXISTS (SELECT *
                FROM joueDans j
                WHERE j.ida <> jb.ida
                AND j.nom LIKE '%Miller%')
GROUP BY a.ida, a.nom, a.prenom
HAVING nb > 1
ORDER BY a.nom, a.prenom DESC;
```

Requêtes multi-sources

Il est possible d'exprimer dans la clause **FROM** une liste de tables séparées par une virgule (,).

SELECT att_1, att_2, att_3, ...
FROM *nomTable*1, *nomTable*2,... ;

- Dans la clause **SELECT** , on spécifie la liste des attributs qui seront retournés dans le résultat. Chaque nom d'attribut est séparé par une virgule (,).
- Dans la clause **FROM** est spécifiée la liste des tables d'où les attributs proviennent. Chaque nom de table est séparé par une virgule (,).
- La requête se termine par un point-virgule (;)

Requêtes multi-sources (2)

Afin de spécifier de quelle table vient quel attribut, il est possible de faire précéder le nom de l'attribut par le nom de la table (avec un point entre les deux).

SELECT $nomTable1.att_1$, $nomTable1.att_2$, $nomTable2.att_3$, ...
FROM $nomTable1$, $nomTable2$,... ;

Définition des alias pour les tables

La notation avec les noms de tables préfixant les attributs peut s'avérer fastidieuse.
Dans la clause **FROM** , il est possible d'associer à chaque instance un alias qui identifiera chaque instance de manière unique.

SELECT A.att_1, A.att_2, B.att_3, ...
FROM *nomTable*1 A, *nomTable*2 B,... ;

- Dans la clause **FROM** l'alias est séparé du nom de la table par un espace ( ).
- Dans la clause **SELECT** , on préfixe les attributs par l'alias
- Il est d'usage d'associer un alias plus court que le nom de la table.

Expression du produit cartésien

SELECT *
FROM *nomTable1*, *nomTable2* ;

- Le produit cartésien permet d'associer à chaque tuple de la première table *nomTable1* chaque tuple de la table *nomTable2*.
- L'utilisation de * dans la clause **SELECT** fait que le résultat est composé de tous les attributs composant les deux tables.
- Les tables *nomTable1* et *nomTable2* peuvent être les mêmes (*i.e.*,deux instances de la même table)

Exemple 1 de Produit cartésien

Donner le produit cartésien d'artiste avec pays

```
SELECT *
FROM Artiste, Pays ;
```

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

ida	prenom	nom	nation	code	nom
1	Jean	DUJARDIN	FR	FR	France
1	Jean	DUJARDIN	FR	USA	Etats-Unis
1	Jean	DUJARDIN	FR	UK	Royaume-Uni
1	Jean	DUJARDIN	FR	AR	Argentine
2	Scarlett	JOHANSSON	USA	FR	France
2	Scarlett	JOHANSSON	USA	USA	Etats-Unis
2	Scarlett	JOHANSSON	USA	UK	Royaume-Uni
2	Scarlett	JOHANSSON	USA	AR	Argentine
3	Tom	CRUISE	USA	FR	France
3	Tom	CRUISE	USA	USA	Etats-Unis
3	Tom	CRUISE	USA	UK	Royaume-Uni
3	Tom	CRUISE	USA	AR	Argentine
4	Bérénice	BEJO	FR-AR	FR	France
...	...	...	...	...	...

Pays(code, nom)

code	nom
FR	France
USA	Etats-Unis
UK	Royaume-Uni
AR	Argentine

Exemple 1 de Produit cartésien

Donner le produit cartésien d'artiste avec pays

```
SELECT *
FROM Artiste, Pays ;
```

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

ida	prenom	nom	nation	code	nom
1	Jean	DUJARDIN	FR	FR	France
1	Jean	DUJARDIN	FR	USA	Etats-Unis
1	Jean	DUJARDIN	FR	UK	Royaume-Uni
1	Jean	DUJARDIN	FR	AR	Argentine
2	Scarlett	JOHANSSON	USA	FR	France
2	Scarlett	JOHANSSON	USA	USA	Etats-Unis
2	Scarlett	JOHANSSON	USA	UK	Royaume-Uni
2	Scarlett	JOHANSSON	USA	AR	Argentine
3	Tom	CRUISE	USA	FR	France
3	Tom	CRUISE	USA	USA	Etats-Unis
3	Tom	CRUISE	USA	UK	Royaume-Uni
3	Tom	CRUISE	USA	AR	Argentine
4	Bérénice	BEJO	FR-AR	FR	France
...	...	...	...	...	...

Pays(code, nom)

code	nom
FR	France
USA	Etats-Unis
UK	Royaume-Uni
AR	Argentine

Exemple 1 de Produit cartésien

Donner le produit cartésien d'artiste avec pays

```
SELECT *
FROM Artiste, Pays ;
```

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

ida	prenom	nom	nation	code	nom
1	Jean	DUJARDIN	FR	FR	France
1	Jean	DUJARDIN	FR	USA	Etats-Unis
1	Jean	DUJARDIN	FR	UK	Royaume-Uni
1	Jean	DUJARDIN	FR	AR	Argentine
2	Scarlett	JOHANSSON	USA	FR	France
2	Scarlett	JOHANSSON	USA	USA	Etats-Unis
2	Scarlett	JOHANSSON	USA	UK	Royaume-Uni
2	Scarlett	JOHANSSON	USA	AR	Argentine
3	Tom	CRUISE	USA	FR	France
3	Tom	CRUISE	USA	USA	Etats-Unis
3	Tom	CRUISE	USA	UK	Royaume-Uni
3	Tom	CRUISE	USA	AR	Argentine
4	Bérénice	BEJO	FR-AR	FR	France
...	...	...	...	...	...

Pays(code, nom)

code	nom
FR	France
USA	Etats-Unis
UK	Royaume-Uni
AR	Argentine

Plusieurs instances... donc risque d'ambiguïté !

Question : Que retourne la requête suivante ?

```
SELECT nom
FROM Artiste, Pays ;
```

Est-ce le nom des artistes ?
Est-ce le nom des pays ?
Est-ce les deux ?
En fait, il y a ambiguïté ! ce qui se traduit par une erreur !
La requête est fausse !

Plusieurs instances... donc risque d'ambiguïté !

Question : Que retourne la requête suivante ?

```
SELECT nom
FROM Artiste, Pays ;
```

Est-ce le nom des artistes ?
Est-ce le nom des pays ?
Est-ce les deux ?
En fait, il y a ambiguïté ! ce qui se traduit par une erreur !
La requête est fausse !

Plusieurs instances... donc risque d'ambiguïté !

Dans ce cas, il est indispensable de préciser la provenance de l'attribut nom.

Donner tous les couples nom d'artiste, nom de pays possibles.

SELECT A.nom **AS** nomArtiste, P.nom **AS** nomPays
FROM *Artiste* A, *Pays* P ;

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

Pays(code, nom)

code	nom
FR	France
USA	Etats-Unis
UK	Royaume-Uni
AR	Argentine

Résultat :

nomArtiste	nomPays
DUJARDIN	France
DUJARDIN	Etats-Unis
DUJARDIN	Royaume-Uni
DUJARDIN	Argentine
JOHANSSON	France
JOHANSSON	Etats-Unis
JOHANSSON	Royaume-Uni
JOHANSSON	Argentine
CRUISE	France
CRUISE	Etats-Unis
CRUISE	Royaume-Uni
CRUISE	Argentine
BEJO	France
...	...

Un raccourci syntaxique pour dire "tous les attributs d'une des tables"

Il est possible de préfixer l'étoile (*) par le nom/alias de la table.

Donner chaque artiste autant de fois qu'il y a de pays

SELECT A.* **FROM** *Artiste* A, *Pays* P ;

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

Pays(code, nom)

code	nom
FR	France
USA	Etats-Unis
UK	Royaume-Uni
AR	Argentine

Résultat :

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
1	Jean	DUJARDIN	FR
1	Jean	DUJARDIN	FR
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
2	Scarlett	JOHANSSON	USA
2	Scarlett	JOHANSSON	USA
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
3	Tom	CRUISE	USA
3	Tom	CRUISE	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
...	...	...	...

Plan

Retour sur le Produit Cartésien

Donner le produit cartésien d'artiste avec pays

```
SELECT *
FROM Artiste, Pays ;
```

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

Résultat :

ida	prenom	nom	nation	code	nom
1	Jean	DUJARDIN	FR	FR	France
1	Jean	DUJARDIN	FR	USA	Etats-Unis
1	Jean	DUJARDIN	FR	UK	Royaume-Uni
1	Jean	DUJARDIN	FR	AR	Argentine
2	Scarlett	JOHANSSON	USA	FR	France
2	Scarlett	JOHANSSON	USA	USA	Etats-Unis
2	Scarlett	JOHANSSON	USA	UK	Royaume-Uni
2	Scarlett	JOHANSSON	USA	AR	Argentine
3	Tom	CRUISE	USA	FR	France
3	Tom	CRUISE	USA	USA	Etats-Unis
3	Tom	CRUISE	USA	UK	Royaume-Uni
3	Tom	CRUISE	USA	AR	Argentine
4	Bérénice	BEJO	FR-AR	FR	France
...	...	...	...	...	...

Pays(code, nom)

code	nom
FR	France
USA	Etats-Unis
UK	Royaume-Uni
AR	Argentine

Besoin de joindre sur plusieurs tables

Pour répondre à certains besoins, il est nécessaire de mettre en relation des attributs issus de tables différentes.

Exemple de besoins portant sur plusieurs tables

- Donner le nom des pays des artistes dont le nom commence par un 'B'.
- Donner le nom et le prénom des artistes ayant joué dans un film réalisé par 'Doug LIMAN'.
- Donner le nom des artistes américain(e)s ayant joué pour un réalisateur français.
- Donner les films ayant plus d'un quart de son budget consacré au salaire d'un(e) artiste.
- ...

Pour répondre à ces besoins, nous allons devoir **joindre des tables**.

Différents types de jointures

Il existe différent types de jointures :

- Jointure interne (INNER JOIN)
  - θ-jointure et équi-jointure
  - Jointure naturelle
  - Semi-jointure
- Jointure externe (OUTTER JOIN)
  - Jointure gauche et Jointure droite
  - Jointure plein

Jointure interne ou θ-jointure

Une jointure interne s'exprime dans la clause **FROM** de la requête. La **jointure interne** est fréquemment utilisée : seuls les tuples qui respectent la condition de jointure sont conservés.

SELECT att_1, att_2, ...
FROM $nomTable_1$ **[INNER] JOIN** $nomTable_2$
ON *condition_de_jointure*
[**WHERE** *conditions_de_selection*] ;

: **Rappel : ce qui est entre crochets ([ ]) est optionnel.**

- La condition de jointure est introduite par **ON** .
- la condition de jointure porte sur un opérateur quelconque ($=$, $!=$, $>$, $>=$, $<$, $=<$, **IN** , **LIKE** , ...) appliqué à deux attributs ou une combinaison (conjonction/disjonction) d'opérateurs appliqués
  NB : on parle aussi de *theta*-**jointure**.

Jointure interne : équi-jointure et auto-jointure

En fonction de la spécificité de la condition de jointure, la jointure peut être qualifiée d'équi-jointure ou d'auto-jointure.

SELECT A.att_1, B2.att_2, ...
FROM $nomTable_1$ A **JOIN**
($nomTable_2$ B1 **JOIN** $nomTable_2$ B2
ON B1.att_x < B2.att_z)
ON A.att_x = B2.att_x ;

Types de conditions dans les jointures :

- la condition de jointure est une égalité ou une conjonction exclusive d'égalités
  $\Rightarrow$ la *theta*-jointure est dite alors **équi-jointure**
- la jointure porte sur une table avec elle-même
  $\Rightarrow$ la *theta*-jointure est dite alors **auto-jointure**

Exemple 1 de jointure interne (équi-jointure)

Donner le nom des personnages du film 'The Artist'

SELECT *JD*.nom
FROM *Film* F **JOIN** *JoueDans* JD **ON** *F*.idF = *JD*.idF
WHERE *Film*.titre = 'The Artist' ;

Film(idf, titre, annee, genre, real, budget)

idf	titre	annee	genre	real	budget
1	Edge of tomorrow	2014	SF	7	178
2	Lucy	2014	SF	8	40
3	The Artist	2011	Drame	9	9
4	OSS 117: Le Caire, Nid d'espions	2006	NULL	9	14
5	Bienvenue chez les Ch'tis	2008	Comédie	5	11

JoueDans(ida, idf, nom)

idp	idf	nom	sal
1	3	George Valentin	0,3
1	4	Hubert Bonisseur de La Bath	0,2
2	2	Lucy Miller	14
3	1	Bill Cage	20
4	4	Larmina El Akmar Betouche	NULL
4	3	Peppy Miller	NULL
5	5	Antoine Bailleul	1,3
6	1	Rita Vrataski	NULL

$\Rightarrow$ Résultat :

nom
George Valentin
Peppy Miller

Exemple 2 de jointure interne (Auto-jointure)

Donner les couples de titres de film de même genre

SELECT *F*1.titre **AS** Film1, *F*2.titre **AS** Film2
FROM *Film* F1 **JOIN** *Film* F2 **ON** *F*1.genre = *F*2.genre ;

Film(idf, titre, annee, genre, real, budget)

idf	titre	annee	genre	real	budget
1	Edge of tomorrow	2014	SF	7	178
2	Lucy	2014	SF	8	40
3	The Artist	2011	Drame	9	9
4	OSS 117: Le Caire, Nid d'espions	2006	NULL	9	14
5	Bienvenue chez les Ch'tis	2008	Comédie	5	11

$\Rightarrow$ Résultat :

Film1	Film2
Edge of tomorrow	Edge of tomorrow
Edge of tomorrow	Lucy
Lucy	Lucy
Lucy	Edge of tomorrow

Exemple 3 de jointure interne (θ-jointure)

Donner le titre des films dont le budget est inférieur ou égal au salaire du personnage 'Lucy Miller'

SELECT *F*.titre
FROM *Film* F **JOIN** *JoueDans* JD **ON** *F*.budget <= *JD*.sal
WHERE *JD*.nom = 'Lucy Miller' ;

Film(idf, titre, annee, genre, real, budget)

idf	titre	annee	genre	real	budget
1	Edge of tomorrow	2014	SF	7	178
2	Lucy	2014	SF	8	40
3	The Artist	2011	Drame	9	9
4	OSS 117: Le Caire, Nid d'espions	2006	NULL	9	14
5	Bienvenue chez les Ch'tis	2008	Comédie	5	11

JoueDans(ida, idf, nom)

idp	idf	nom	sal
1	3	George Valentin	0,3
1	4	Hubert Bonisseur de La Bath	0,2
2	2	Lucy Miller	14
3	1	Bill Cage	20
4	4	Larmina El Akmar Betouche	NULL
4	3	Peppy Miller	NULL
5	5	Antoine Bailleul	1,3
6	1	Rita Vrataski	NULL

$\Rightarrow$ Résultat :

titre
The Artist
OSS 117 : Le Caire, Nid d'espions
Bienvenue chez les Ch'tis

Remarque sur l'exemple 2 de jointure interne (1)

On remarque que le résultat retourné n'est pas satisfaisant, du fait des doublons.

Intuitivement, il faut que les deux films d'un tuple résultat soient différents.

Dans ce cas la requête devient :

Donner les couples de titres de film de même genre

SELECT *F*1.titre **AS** Film1, *F*2.titre **AS** Film2
FROM *Film* F1 **JOIN** *Film* F2 **ON** *F*1.genre = *F*2.genre
AND *F*1.idf != *F*2.idf ;

Mais le résultat sera :

Résultat :

Film1	Film2
Edge of tomorrow	Lucy
Lucy	Edge of tomorrow

Remarque sur l'exemple 2 de jointure interne (2)

Pour obtenir le résultat attendu, il est nécessaire de n'avoir que (Edge of tomorrow, Lucy) ou (Lucy, Edge of tomorrow) dans le résultat, mais pas les deux.

L'astuce consiste à ajouter une condition de sélection afin de prendre que le tuple dont l'identifiant Film1 est plus petit que l'identifiant de Film2.

La requête devient :

Donner les couples de titres de film de même genre

SELECT *F*1.titre **AS** Film1, *F*2.titre **AS** Film2
FROM *Film* F1 **JOIN** *Film* F2 **ON** *F*1.genre = *F*2.genre
AND *F*1.idf < *F*2.idf ;

Et le résultat devient :

Résultat :

Film1	Film2
Edge of tomorrow	Lucy

Jointure interne *'Old-School'*

Même si la syntaxe est obsolète, il est intéressant de remarquer qu'une jointure interne peut s'exprimer comme la combinaison d'un produit cartésien et d'une sélection.
Ainsi l'expression d'un **JOIN ... ON**

SELECT att_1, att_2, ...
FROM $nomTable_1$ **[INNER] JOIN** $nomTable_2$
ON *condition_de_jointure*
[**WHERE** *conditions_de_selection*] ;

peut s'exprimer de la manière suivante :

SELECT att_1, att_2, ...
FROM $nomTable_1$, $nomTable_2$
WHERE *condition_de_jointure*
[**AND** *conditions_de_selection*] ;

Retour sur l'exemple 2 de jointure interne

Donner le nom des personnages du film 'The Artist'

```
SELECT JD.nom
FROM Film F JOIN JoueDans JD ON F.idF = JD.idF
WHERE Film.titre = 'The Artist' ;
```

Donner le nom des personnages du film 'The Artist'

```
SELECT JD.nom
FROM Film F, JoueDans JD
WHERE F.idF = JD.idF
AND Film.titre = 'The Artist' ;
```

Expression de la jointure naturelle (1)

Nous remarquons que sur la requête suivante, nous avons une équi-jointure et que l'égalité porte sur tous les attributs communs aux deux tables (*i.e.*, *idF*).

Donner le nom des personnages du film 'The Artist'

```
SELECT JD.nom
FROM Film F JOIN JoueDans JD ON F.idF = JD.idF
WHERE Film.titre = 'The Artist';
```

Un raccourci syntaxique possible est d'utiliser une jointure naturelle.

Donner le nom des personnages du film 'The Artist'

```
SELECT JD.nom
FROM Film F NATURAL JOIN JoueDans JD
WHERE Film.titre = 'The Artist';
```

Expression de la jointure naturelle (2)

La jointure naturelle permet d'exprimer une équi-jointure pour laquelle la condition de jointure porte sur l'ensemble des attributs communs aux deux tables jointes.

SELECT att_1, att_2, ...
FROM $nomTable_1$ A **NATURAL JOIN** $nomTable_2$;

ce qui correspond à :

SELECT att_1, att_2, ...
FROM $nomTable_1$ A **JOIN** $nomTable_2$ B
ON A.att_x = B.att_x **AND** A.att_y = B.att_y
AND A.att_z = B.att_z ;

- avec $\{att_x, att_y, att_z\}$ l'ensemble des attributs communs aux tables $nomTable_1$ et $nomTable_2$.

Exemple 1 de jointure naturelle

Pour chaque nom de personnage, donner le titre du film dans lequel ledit personnage apparaît

SELECT *JD*.nom, *F*.titre
FROM *JoueDans* JD **NATURAL JOIN** *Film* F ;

Film(idf, titre, annee, genre, real, budget)

idf	titre	annee	genre	real	budget
1	Edge of tomorrow	2014	SF	7	178
2	Lucy	2014	SF	8	40
3	The Artist	2011	Drame	9	9
4	OSS 117: Le Caire, Nid d'espions	2006	NULL	9	14
5	Bienvenue chez les Ch'tis	2008	Comédie	5	11

⇒

JoueDans(ida, idf, nom)

idp	idf	nom	sal
1	3	George Valentin	0,3
1	4	Hubert Bonisseur de La Bath	0,2
2	2	Lucy Miller	14
3	1	Bill Cage	20
4	4	Larmina El Akmar Betouche	NULL
4	3	Peppy Miller	NULL
5	5	Antoine Bailleul	1,3
6	1	Rita Vrataski	NULL

Résultat :

nom	titre
George Valentin	The Artist
Hubert Bonisseur de La Bath	OSS 117 : ...
Lucy Miller	Lucy
Bill Cage	Edge of tomorrow
Larmina El Akmar Betouche	OSS 117 : ...
Peppy Miller	The Artist
Antoine Bailleul	Bienvenue chez ...
Rita Vrataski	Edge of tomorro

Question à propos de la jointure naturelle

Que retourne la requête suivante ?

???

```
SELECT A.nom, JD.sal
 FROM JoueDans JD NATURAL JOIN Artiste A ;
```

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

JoueDans(ida, idf, nom)

idp	idf	nom	sal
1	3	George Valentin	0,3
1	4	Hubert Bonisseur de La Bath	0,2
2	2	Lucy Miller	14
3	1	Bill Cage	20
4	4	Larmina El Akmar Betouche	NULL
4	3	Peppy Miller	NULL
5	5	Antoine Bailleul	1,3
6	1	Rita Vrataski	NULL

Réponse

Donner le nom des artistes qui ont joué dans un film un personnage dont le prénom et le nom correspond au nom dudit artiste

```
SELECT A.nom, JD.sal
 FROM JoueDans JD NATURAL JOIN Artiste A ;
```

La requête retournera :

nom	sal

car elle est équivalente à :

```
SELECT A.nom, JD.sal
FROM JoueDans JD JOIN Artiste A
ON JD.idA = A.idA AND JD.nom = A.nom ;
```

Expression de la semi-jointure

Une requête dans laquelle on effectue une jointure et que le résultat retourné ne porte que sur les attributs d'une des deux tables exprime une **semi-jointure** :

SELECT $nomTable_2$.*
FROM $nomTable_1$ **JOIN** $nomTable_2$
ON $nomTable_1.att_x = nomTable_2.att_x$;

Exemple de semi-jointure

Donner les artistes qui n'ont pas de multi-nationalité

SELECT *A*.*
FROM *Artiste* A **JOIN** *Pays* **PON** *A*.nation = *P*.code ;

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

Pays(code, nom)

code	nom
FR	France
USA	Etats-Unis
UK	Royaume-Uni
AR	Argentine

$\Rightarrow$

Résultat :

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

Remarque

Donner pour chaque artiste son prénom et son nom et le nom de son pays d'origine

```
SELECT A.prenom,A.nom, P.nom AS pays
 FROM Artiste A JOIN Pays P
 ON A.nation = P.code;
```

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

Pays(code, nom)

code	nom
FR	France
USA	Etats-Unis
UK	Royaume-Uni
AR	Argentine

$\Rightarrow$

Résultat :

prenom	nom	pays
Jean	DUJARDIN	France
Scarlett	JOHANSSON	Etats-Unis
Tom	CRUISE	Etats-Unis
Dany	BOON	France
Emily	BLUNT	Royaume-uni
Doug	LIMAN	Etats-Unis
Luc	BESSON	Fance
Michel	HAZANAVICIUS	France

Nous n'avons plus "chaque artiste" !

Il faudra s'assurer de garder chaque artiste et si son pays d'origine est dans la table Pays alors, on associe le nom du pays, sinon l'attribut reste sans valeur.

Expression d'une jointure externe : OUTER JOIN

Une requête avec jointure externe (**OUTER JOIN**) affiche un sur-ensemble des tuples qui vérifient la condition de la jointure. Les tuples qui vérifient la condition de jointure, ont des valeurs issues des deux tables jointes. Les autres tuples, qui ne vérifient pas la condition de jointure, ont les attributs issus d'une des deux tables sans valeur (*i.e.*, à **NULL**).
il existe trois types de jointure externes :

- la jointure gauche (**LEFT JOIN**)
- la jointure droite (**RIGHT JOIN**)
- la jointure pleine (**FULL JOIN**)

SELECT *
FROM $nomTable_1$ <**LEFT**|**RIGHT**|**FULL**> [**OUTER**] **JOIN** $nomTable_2$
[**WHERE** *conditions_de_selection*];

Différences entre **LEFT JOIN** , **RIGHT JOIN** et **FULL JOIN**

- **LEFT JOIN** : les tuples de la table de gauche sans correspondance dans l'autre table sont incluses dans le résultat avec une valeur **NULL** pour les attributs de l'autre table
- **RIGHT JOIN** : les tuples de la table de droite sans correspondance dans l'autre table sont incluses dans le résultat avec une valeur **NULL** pour les attributs de l'autre table
- **FULL JOIN** : toutes les lignes de chacune des tables sont retournées. Les lignes sans correspondance ont leurs attributs complétés par la valeur **NULL** .

Exemple de jointure externe gauche

Donner pour chaque artiste son prénom et son nom et le nom de son pays d'origine

```
SELECT A.prenom,A.nom, P.nom AS pays
 FROM Artiste A LEFT JOIN Pays P
 ON A.nation = P.code;
```

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

Pays(code, nom)

code	nom
FR	France
USA	Etats-Unis
UK	Royaume-Uni
AR	Argentine

$\Rightarrow$

Résultat :

prenom	nom	pays
Jean	DUJARDIN	France
Scarlett	JOHANSSON	Etats-Unis
Tom	CRUISE	Etats-Unis
Bérénice	BEJO	NULL
Dany	BOON	France
Emily	BLUNT	Royaume-uni
Doug	LIMAN	Etats-Unis
Luc	BESSON	France
Michel	HAZANAVICIUS	France

Nous avons bien "chaque artiste" !

Exemple de jointure externe droite

Donner pour chaque pays son nom, ainsi que le prénom et le nom des artistes originaires dudit pays.

```
SELECT P.nom AS pays, A.prenom,A.nom
 FROM Artiste A RIGHT JOIN Pays P
 ON A.nation = P.code;
```

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

Pays(code, nom)

code	nom
FR	France
USA	Etats-Unis
UK	Royaume-Uni
AR	Argentine

$\Rightarrow$

Résultat :

pays	prenom	nom
France	Jean	DUJARDIN
France	Dany	BOON
France	Luc	BESSON
France	Michel	HAZANAVICIUS
Etats-Unis	Scarlett	JOHANSSON
Etats-Unis	Tom	CRUISE
Etats-Unis	Doug	LIMAN
Royaume-uni	Emily	BLUNT
Argentine	NULL	NULL

Nous avons bien "chaque pays" !

Exemple précédent exprimé par une jointure gauche

Donner pour chaque pays son nom, ainsi que le prénom et le nom des artistes originaires dudit pays.

```
SELECT P.nom AS pays, A.prenom,A.nom
 FROM Pays P LEFT JOIN Artiste A
 ON A.nation = P.code;
```

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

Pays(code, nom)

code	nom
FR	France
USA	Etats-Unis
UK	Royaume-Uni
AR	Argentine

$\Rightarrow$

Résultat :

pays	prenom	nom
France	Jean	DUJARDIN
France	Dany	BOON
France	Luc	BESSON
France	Michel	HAZANAVICIUS
Etats-Unis	Scarlett	JOHANSSON
Etats-Unis	Tom	CRUISE
Etats-Unis	Doug	LIMAN
Royaume-uni	Emily	BLUNT
Argentine	NULL	NULL

Exemple de jointure externe pleine

Donner pour chaque artiste son prénom et son nom et son pays d'origine, sans perdre de pays.

```
SELECT A.prenom,A.nom, P.nom AS pays
 FROM Artiste A FULL JOIN Pays P
 ON A.nation = P.code;
```

Artiste(ida, nom, prenom, nation)

ida	prenom	nom	nation
1	Jean	DUJARDIN	FR
2	Scarlett	JOHANSSON	USA
3	Tom	CRUISE	USA
4	Bérénice	BEJO	FR-AR
5	Dany	BOON	FR
6	Emily	BLUNT	UK
7	Doug	LIMAN	USA
8	Luc	BESSON	FR
9	Michel	HAZANAVICIUS	FR

Pays(code, nom)

code	nom
FR	France
USA	Etats-Unis
UK	Royaume-Uni
AR	Argentine

$\Rightarrow$

Résultat :

prenom	nom	pays
Jean	DUJARDIN	France
Scarlett	JOHANSSON	Etats-Unis
Tom	CRUISE	Etats-Unis
Bérénice	BEJO	NULL
Dany	BOON	France
Emily	BLUNT	Royaume-uni
Doug	LIMAN	Etats-Unis
Luc	BESSON	France
Michel	HAZANAVICIUS	France
NULL	NULL	Argentine

Nous avons bien "chaque artiste" et tous les pays apparaissent dans la table !

Bilan sur les jointures

- Jointure interne : **JOIN ... ON**
  C'est la jointure la plus courante qui retourne uniquement les tuples qui respectent la condition de jointure.
- Jointure naturelle : **NATURAL JOIN**
  C'est un raccourci syntaxique qui permet d'exprimer une équi-jointure sur tous des attributs communs aux deux tables jointes.
- Jointure externe : **LEFT JOIN** , **RIGHT JOIN** et **FULL JOIN**
  Ce sont des jointures qui retournent non seulement les tuples qui respectent la condition de jointure, mais aussi des tuples issus d'une des deux tables (ou des deux) complétés par des attributs sans valeurs issues de l'autre table.