

Séance à distance

SQL : interroger plusieurs tables avec JOIN

— Version corrigée —

Déroulé conseillé : Partie 1 (pourquoi plusieurs tables, ≈ 10 min), Partie 2 (lecture des exemples, ≈ 25 min), exercice 1 sur papier (≈ 25 min), exercice 2 sur machine (≈ 50 min).

1 Pourquoi plusieurs tables ?

Dans la séance précédente, la table `livres` contenait une colonne `auteur` sous forme de texte. Imaginons que l'on veuille aussi conserver la nationalité et l'année de naissance de chaque auteur :

titre	auteur	nationalite	naissance
Les Misérables	Victor Hugo	française	1802
Notre-Dame de Paris	Victor Hugo	française	1802
Quatrevingt-treize	V. Hugo	française	1820

Les informations sur Victor Hugo sont **répétées** pour chacun de ses livres. C'est encombrant, et surtout dangereux : à la troisième ligne, une faute de frappe rend la base **incohérente** (Hugo est-il né en 1802 ou en 1820 ?). Pour éviter cela, on range les auteurs dans **leur propre table**, et chaque livre indique seulement le *numéro* de son auteur.

Définition : clé étrangère

Une **clé étrangère** est une colonne d'une table dont les valeurs sont des valeurs de la **clé primaire d'une autre table**. Elle sert à relier une ligne à une ligne d'une autre table. Ci-dessous, la colonne `id_auteur` de la table `livres` est une clé étrangère qui fait référence à la colonne `id` de la table `auteurs`. Ainsi, le livre *Germinal* a `id_auteur = 3` : son auteur est la ligne de `auteurs` dont l'`id` vaut 3, c'est-à-dire Émile Zola.

La base bibliotheque

La bibliothèque a donc réorganisé sa base en quatre tables. Les clés primaires sont soulignées, les clés étrangères sont marquées d'un #.

Table auteurs

<u>id</u>	nom	prenom	nationalite	naissance
1	Hugo	Victor	française	1802
2	Verne	Jules	française	1828
3	Zola	Émile	française	1840
4	Camus	Albert	française	1913
5	Shelley	Mary	britannique	1797
6	Orwell	George	britannique	1903
7	Dumas	Alexandre	française	1802
8	Saint-Exupéry	Antoine de	française	1900
9	Christie	Agatha	britannique	1890
10	Proust	Marcel	française	1871

Table adherents

<u>id</u>	nom	prenom	ville
1	Martin	Léa	Saint-Étienne
2	Bernard	Hugo	Firminy
3	Petit	Sarah	Saint-Étienne
4	Durand	Tom	Roanne
5	Moreau	Inès	Firminy

Table livres

<u>id</u>	titre	annee	genre	#id_auteur
1	Les Misérables	1862	roman	1
2	Notre-Dame de Paris	1831	roman	1
3	Vingt mille lieues sous les mers	1870	aventure	2
4	Le Tour du monde en 80 jours	1872	aventure	2
5	Germinal	1885	roman	3
6	L'Assommoir	1877	roman	3
7	L'Étranger	1942	roman	4
8	La Peste	1947	roman	4
9	Frankenstein	1818	science-fiction	5
10	1984	1949	science-fiction	6
11	La Ferme des animaux	1945	roman	6
12	Le Comte de Monte-Cristo	1844	aventure	7
13	Les Trois Mousquetaires	1844	aventure	7
14	Le Petit Prince	1943	conte	8
15	Le Crime de l'Orient-Express	1934	policier	9
16	Mort sur le Nil	1937	policier	9

Table emprunts

<u>id</u>	#id_livre	#id_adherent	date_emprunt
1	14	1	2026-09-02
2	7	1	2026-09-02
3	3	2	2026-09-05
4	10	3	2026-09-08
5	14	3	2026-09-10
6	1	4	2026-09-12
7	15	2	2026-09-15
8	8	1	2026-09-20
9	12	4	2026-09-22
10	10	2	2026-09-25
11	16	3	2026-09-28
12	4	4	2026-10-01

Dans la table **emprunts**, chaque ligne indique *quel livre* (`id_livre`, qui fait référence à `livres.id`) a été emprunté par *quel adhérent* (`id_adherent`, qui fait référence à `adherents.id`) et à quelle date. Les dates sont écrites sous la forme 'AAAA-MM-JJ'.

2 La jointure JOIN ... ON ... en exemples

Les tables sont maintenant bien rangées... mais pour afficher « le titre d'un livre et le nom de son auteur », il faut aller chercher l'information dans **deux** tables à la fois. C'est le rôle de la **jointure**.

Définition : jointure

La requête

```
SELECT ... FROM table1 JOIN table2 ON condition;
```

associe chaque ligne de **table1** à chaque ligne de **table2** **pour laquelle la condition écrite après ON est vraie**. Chaque association produit une ligne du résultat, qui contient toutes les colonnes des deux tables. En pratique, la condition est presque toujours de la forme

clé étrangère = clé primaire correspondante.

Lorsque deux tables sont en jeu, on désigne une colonne par `table.colonne` (par exemple `livres.titre` ou `auteurs.id`).

Exemple 1 : une première jointure

```
SELECT livres.titre, auteurs.nom
FROM livres JOIN auteurs ON livres.id_auteur = auteurs.id;
```

Résultat :

titre	nom
Les Misérables	Hugo
Notre-Dame de Paris	Hugo
Vingt mille lieues sous les mers	Verne
...	... (16 lignes au total)

Comment lire cette requête ? Pour chaque livre, le SGBD cherche dans **auteurs** la (ou les)

ligne(s) dont l'id est égal à l'id_auteur du livre, et les « recolle » ensemble. Par exemple, pour *Germinal* :

ligne de livres					→	ligne de auteurs				
5	Germinal	1885	roman	3		3	Zola	Émile	française	1840

On obtient la ligne « Germinal | Zola ». Un même auteur peut donc apparaître plusieurs fois dans le résultat (une fois par livre).

Exemple 2 : pourquoi écrire table.colonne ?

Les tables livres et auteurs ont toutes les deux une colonne id. Si l'on écrit :

```
SELECT id, titre, nom
FROM livres JOIN auteurs ON livres.id_auteur = auteurs.id;
```

le SGBD répond par une erreur : `ambiguous column name: id`. Il ne sait pas de quel id on parle ! Il faut écrire `livres.id` ou `auteurs.id`.

Conseil : dans une requête avec JOIN, préfixez *toujours* les colonnes par le nom de leur table, même quand ce n'est pas obligatoire. La requête est plus lisible et on évite les erreurs.

Exemple 3 : jointure, WHERE et ORDER BY

Toutes les clauses déjà connues s'utilisent comme avant, après la jointure. Livres écrits par des auteurs britanniques, du plus ancien au plus récent :

```
SELECT livres.titre, livres.annee
FROM livres JOIN auteurs ON livres.id_auteur = auteurs.id
WHERE auteurs.nationalite = 'britannique'
ORDER BY livres.annee;
```

Résultat :

titre	annee
Frankenstein	1818
Le Crime de l'Orient-Express	1934
Mort sur le Nil	1937
La Ferme des animaux	1945
1984	1949

La condition porte sur une colonne de `auteurs`, alors que l'on affiche des colonnes de `livres` : c'est exactement ce que permet la jointure.

Exemple 4 : jointure et COUNT(*)

Combien de livres de la bibliothèque ont été écrits par un auteur français ?

```
SELECT COUNT(*) AS nb_francais
FROM livres JOIN auteurs ON livres.id_auteur = auteurs.id
WHERE auteurs.nationalite = 'française';
```

Résultat :

nb_francais
11

COUNT(*) compte les lignes du résultat de la jointure, après le filtre WHERE.

Exemple 5 : enchaîner deux jointures

On peut joindre plus de deux tables en écrivant plusieurs `JOIN ... ON ...` à la suite. Qui, parmi les adhérents de Roanne, a emprunté quel livre, et quand ?

```
SELECT adherents.prenom, livres.titre, emprunts.date_emprunt
FROM emprunts
JOIN livres ON emprunts.id_livre = livres.id
JOIN adherents ON emprunts.id_adherent = adherents.id
WHERE adherents.ville = 'Roanne';
```

Résultat :

prenom	titre	date_emprunt
Tom	Les Misérables	2026-09-12
Tom	Le Comte de Monte-Cristo	2026-09-22
Tom	Le Tour du monde en 80 jours	2026-10-01

On part de la table `emprunts`, qui fait le lien : chaque emprunt est recollé à son livre (première jointure) puis à son adhérent (seconde jointure). Il y a **une condition ON par JOIN**.

À retenir

- Chaque **JOIN** est **toujours** suivi de sa condition **ON**, qui relie en général une clé étrangère à la clé primaire qu'elle référence.
- Une ligne qui ne trouve **aucune** correspondance dans l'autre table n'apparaît pas dans le résultat.
- Les clauses s'écrivent toujours dans cet ordre :

```
SELECT ... FROM ... JOIN ... ON ... JOIN ... ON ...
WHERE ... ORDER BY ... LIMIT ... OFFSET ...;
```

3 Exercices

Exercice 1 : à la main, sans machine

On utilise les tables de la partie 1. Pour les questions qui demandent un résultat, l'ordre des lignes n'a pas d'importance.

1. Sans écrire de requête : qui est l'auteur de *1984* ? Expliquer comment on le trouve à partir des tables.
2. Donner le résultat de la requête suivante :

```
SELECT livres.titre, auteurs.prenom
FROM livres JOIN auteurs ON livres.id_auteur = auteurs.id
WHERE livres.genre = 'policier';
```

3. Donner le résultat de la requête suivante. Pourquoi certains noms apparaissent-ils deux fois ?

```
SELECT auteurs.nom
FROM livres JOIN auteurs ON livres.id_auteur = auteurs.id
WHERE livres.annee > 1940;
```

4. Dans le résultat de l'exemple 1, Marcel Proust apparaît-il ? Pourquoi ?
5. Que fait la requête suivante ? Donner son résultat, puis la décrire par une phrase en français.

```
SELECT adherents.nom, emprunts.date_emprunt
FROM emprunts JOIN adherents ON emprunts.id_adherent = adherents.id
WHERE emprunts.id_livre = 10;
```

6. Un élève veut afficher chaque titre avec le nom de son auteur. Il écrit :

```
SELECT livres.titre, auteurs.nom
FROM livres JOIN auteurs ON livres.id = auteurs.id;
```

La requête s'exécute sans erreur.

- Donner les trois premières lignes du résultat. Combien de lignes le résultat contient-il ?
- Quelle est l'erreur ? Corriger la requête.

Correction

1. Dans livres, 1984 a id_auteur = 6. Dans auteurs, la ligne d'id 6 est George Orwell. C'est exactement ce que fait une jointure.

	titre	premier
2.	Le Crime de l'Orient-Express	Agatha
	Mort sur le Nil	Agatha

3. Livres parus après 1940 : *L'Étranger*, *La Peste*, *1984*, *La Ferme des animaux*, *Le Petit Prince*. Le résultat contient 5 lignes : Camus, Camus, Orwell, Orwell, Saint-Exupéry. Le résultat contient une ligne **par livre** : Camus et Orwell ont chacun écrit deux de ces livres.

4. Non : aucun livre n'a id_auteur = 10. La ligne de Proust ne trouve aucune correspondance, elle n'apparaît donc pas dans le résultat de la jointure.

	nom	date_emprunt
5. Le livre d'id 10 est 1984 (emprunts 4 et 10).	Petit	2026-09-08
	Bernard	2026-09-25

La requête affiche le nom des adhérents qui ont emprunté 1984, avec la date de chaque emprunt.

6.a) La condition associe le livre n°k à l'auteur n°k : « Les Misérables | Hugo », « Notre-Dame de Paris | Verne », « Vingt mille lieues sous les mers | Zola ». Il y a 10 auteurs, donc seuls les livres d'id 1 à 10 trouvent une correspondance : **10 lignes** (la dernière est « 1984 | Proust »!).

6.b) On a comparé deux clés primaires qui n'ont aucun rapport : le numéro d'un livre n'est pas le numéro de son auteur. Il faut relier la clé étrangère à la clé primaire qu'elle référence :

```
SELECT livres.titre, auteurs.nom
FROM livres JOIN auteurs ON livres.id_auteur = auteurs.id;
```

Attention : une requête qui s'exécute sans erreur n'est pas forcément juste !

Où travailler ? Ouvrir l'activité SQL : <https://capitale2.ac-paris.fr/web/c/9fc3-12045167/mcerhttps://capitale2.ac-paris.fr/web/c/9fc3-12045167/mcer>.

Exercice 2 : interroger la bibliothèque

Pour chaque question, écrire la requête SQL, l'exécuter et noter le résultat obtenu. **On n'utilisera jamais directement un numéro id dans un WHERE** : on part toujours du nom, du prénom ou du titre.

- Afficher le titre de chaque livre avec le nom de son auteur.
- Afficher le titre de chaque livre avec le prénom et le nom de son auteur, triés par nom d'auteur.
- Afficher les titres des livres écrits par Albert Camus.
- Afficher le titre, le prénom et le nom de l'auteur des livres d'aventure.

5. Afficher les titres des livres dont l'auteur est né avant 1850, ainsi que le nom de l'auteur.
6. Combien la bibliothèque possède-t-elle de livres écrits par des auteurs britanniques ? Nommer la colonne `nb_britanniques`.
7. Afficher le titre, le nom de l'auteur et l'année du livre le plus récent écrit par un auteur français.
8. Afficher les dates auxquelles *Le Petit Prince* a été emprunté.
9. Combien d'emprunts Sarah Petit a-t-elle effectués ?
10. Afficher les titres et les dates d'emprunt des livres empruntés par Léa Martin, du plus ancien au plus récent emprunt. (*Il faut trois tables.*)
11. (*Bonus*) Afficher le prénom et le nom des auteurs dont un livre a été emprunté par l'adhérent Hugo Bernard. (*Il faut quatre tables, et attention à ne pas confondre l'adhérent Hugo avec l'auteur Hugo !*)
12. (*Bonus*) On peut faire des calculs dans le `SELECT`, par exemple `livres.annee - auteurs.naissance`. Quel livre a été publié par l'auteur le plus jeune au moment de la publication ? À quel âge ?

Correction

```
-- 1. 16 lignes
SELECT livres.titre, auteurs.nom
FROM livres JOIN auteurs ON livres.id_auteur = auteurs.id;

-- 2. 16 lignes, de Camus à Zola
SELECT livres.titre, auteurs.prenom, auteurs.nom
FROM livres JOIN auteurs ON livres.id_auteur = auteurs.id
ORDER BY auteurs.nom;

-- 3. L'Étranger, La Peste
SELECT livres.titre
FROM livres JOIN auteurs ON livres.id_auteur = auteurs.id
WHERE auteurs.prenom = 'Albert' AND auteurs.nom = 'Camus';

-- 4. Vingt mille lieues..., Le Tour du monde... (Jules Verne),
-- Le Comte de Monte-Cristo, Les Trois Mousquetaires (Alexandre Dumas)
SELECT livres.titre, auteurs.prenom, auteurs.nom
FROM livres JOIN auteurs ON livres.id_auteur = auteurs.id
WHERE livres.genre = 'aventure';

-- 5. 9 livres : ceux de Shelley, Hugo, Dumas, Verne et Zola
SELECT livres.titre, auteurs.nom
FROM livres JOIN auteurs ON livres.id_auteur = auteurs.id
WHERE auteurs.naissance < 1850;

-- 6. 5
SELECT COUNT(*) AS nb_britanniques
FROM livres JOIN auteurs ON livres.id_auteur = auteurs.id
WHERE auteurs.nationalite = 'britannique';

-- 7. La Peste, Camus, 1947
SELECT livres.titre, auteurs.nom, livres.annee
FROM livres JOIN auteurs ON livres.id_auteur = auteurs.id
WHERE auteurs.nationalite = 'française'
ORDER BY livres.annee DESC LIMIT 1;

-- 8. 2026-09-02, 2026-09-10
SELECT emprunts.date_emprunt
FROM emprunts JOIN livres ON emprunts.id_livre = livres.id
WHERE livres.titre = 'Le Petit Prince';

-- 9. 3
SELECT COUNT(*)
FROM emprunts JOIN adherents ON emprunts.id_adherent = adherents.id
WHERE adherents.prenom = 'Sarah' AND adherents.nom = 'Petit';

-- 10. Le Petit Prince et L'Étranger (2026-09-02), La Peste (2026-09-20)
SELECT livres.titre, emprunts.date_emprunt
```

```

FROM emprunts
JOIN livres    ON emprunts.id_livre    = livres.id
JOIN adherents ON emprunts.id_adherent = adherents.id
WHERE adherents.prenom = 'Léa' AND adherents.nom = 'Martin'
ORDER BY emprunts.date_emprunt;
-- 11. Jules Verne, Agatha Christie, George Orwell
SELECT auteurs.prenom, auteurs.nom
FROM emprunts
JOIN livres    ON emprunts.id_livre    = livres.id
JOIN auteurs   ON livres.id_auteur     = auteurs.id
JOIN adherents ON emprunts.id_adherent = adherents.id
WHERE adherents.prenom = 'Hugo' AND adherents.nom = 'Bernard';
-- 12. Frankenstein : Mary Shelley avait 21 ans (1818 - 1797)
SELECT livres.titre, livres.annee - auteurs.naissance AS age
FROM livres JOIN auteurs ON livres.id_auteur = auteurs.id
ORDER BY age LIMIT 1;

```