



### Exercice Bases de données relationnelles et SQL

Dans cet exercice, on pourra utiliser les clauses du langage SQL :

SELECT, FROM, WHERE, AND, OR, DISTINCT, ORDER BY, JOIN ... ON, GROUP BY, HAVING, INSERT INTO, UPDATE.

Conventions : clé primaire pour les clés primaires; attributs précédés de # pour les **clés étrangères**.

On considère la base de données d'une **médiathèque**. Schéma relationnel :

- AUTEUR(id\_auteur, nom, prenom, nationalite)
- LIVRE(id\_livre, titre, annee, genre, #id\_auteur)
- ADHERENT(id\_adh, nom, prenom, ville)
- EMPRUNT(id\_emprunt, #id\_adh, #id\_livre, date\_emprunt, date\_retour)

#### Extraits de tables

AUTEUR

<u>id_auteur</u>	nom	prenom	nationalite
1	<i>Herbert</i>	<i>Frank</i>	<i>US</i>
2	<i>Gavaldà</i>	<i>Anna</i>	<i>FR</i>
3	<i>Orwell</i>	<i>George</i>	<i>UK</i>

LIVRE

<u>id_livre</u>	titre	annee	genre	#id_auteur
10	<i>Dune</i>	1965	SF	1
11	<i>Ensemble, c'est...</i>	2002	Roman	2
12	<i>1984</i>	1949	Dystopie	3

ADHERENT

<u>id_adh</u>	nom	prenom	ville
100	<i>Martin</i>	<i>Alice</i>	<i>Paris</i>
101	<i>Dupont</i>	<i>La</i>	<i>Lyon</i>
102	<i>Silva</i>	<i>Hugo</i>	<i>Marseille</i>

EMPRUNT

<u>id_emprunt</u>	#id_adh	#id_livre	date_emprunt	date_retour
1000	100	10	2024-10-01	2024-10-15
1001	101	12	2024-10-03	NULL
1002	100	11	2024-11-05	2024-11-20

**Partie A (4 points)**

1. Expliquer précisément ce qu'est une clé primaire. Indiquer ensuite la clé primaire de la table ADHERENT et justifier ce choix.

**Corrigé**

**Définition.** Attribut (ou ensemble minimal d'attributs) non nul et sans doublon qui identifie de manière unique chaque ligne d'une relation.

**Application.** Dans ADHERENT, la clé primaire est id\_adh (identifiant unique).

**Barème (2 pts).** 1 pt définition (unicité + non-nullité); 1 pt application justifiée.

2. Expliquer précisément ce qu'est une clé étrangère. Préciser le rôle de l'intégrité référentielle entre EMPRUNT et LIVRE. Donner un exemple d'incohérence interdite.

**Corrigé**

**Définition.** Une clé étrangère est un attribut dont les valeurs doivent correspondre à une clé primaire d'une autre relation.

**Référence.** EMPRUNT.#id\_livre référence LIVRE.id\_livre: chaque emprunt doit pointer un livre existant; les suppressions/mises à jour sont encadrées (p. ex. RESTRICT/CASCADE).

**Exemple interdit.** Insérer dans EMPRUNT une ligne avec #id\_livre=9999 sans id\_livre=9999 dans LIVRE.

**Barème (2 pts).** 1 pt définition claire; 1 pt intégrité référentielle + exemple pertinent.

**Partie B Requêtes à expliquer (4 points)**

3. Expliquer la requête suivante et donner le résultat attendu sur l'extrait fourni.

```
SELECT titre
FROM LIVRE
WHERE genre = 'SF'
ORDER BY titre ASC;
```

**Corrigé**

On va dans la table LIVRE, chercher les films de genre ' SF ' .

La sortie triée par ordre alphabétique croissant.

Sur l'extrait on obtient : *Dune*.

**Barème (2 pts).** 1 pt explication filtre/tri; 1 pt résultat attendu.

4. Expliquer la requête suivante et donner le résultat attendu sur l'extrait fourni.

```
SELECT L.titre, A.nom, A.prenom
FROM LIVRE AS L
JOIN AUTEUR AS A ON A.id_auteur = L.id_auteur
WHERE L.annee < 1960;
```



### Corrigé

La jointure relie LIVRE à AUTEUR via id\_auteur; la condition temporelle garde les livres publiés avant 1960.

Sur l'extrait des tables on obtient : ('1984', 'Orwell', 'George').

**Barème (2 pts).** 1 pt explication de la jointure; 1 pt rôle du filtre + résultat.

## Partie C Requetes à écrire (6 points)

5. Ajouter dans AUTEUR l'écrivain Isaac Asimov, nationalité 'US', avec l'identifiant 4. Donner la commande SQL.



### Corrigé

INSERT INTO avec colonnes explicites; id\_auteur unique.

**Barème (2 pts).** 1 pt structure INSERT; 1 pt valeurs/quotes correctes.

```
INSERT INTO AUTEUR (id_auteur, nom, prenom, nationalite)
VALUES (4, 'Asimov', 'Isaac', 'US');
```

6. Une erreur s'est glissée : l'adhérent id\_adh=102 a été enregistré avec la ville 'Marseille' au lieu de 'Paris'. Donner la commande SQL permettant de corriger cette erreur.



### Corrigé

UPDATE ciblé par WHERE; attention aux quotes pour 'Paris'.

**Barème (2 pts).** 1 pt SET; 1 pt WHERE.

```
UPDATE ADHERENT
SET ville = 'Paris'
WHERE id_adh = 102;
```

7. Écrire une requête SQL qui renvoie les titre des livres de genre 'Roman', triés par année décroissante, puis par titre croissant.



### Corrigé

Filtrer genre='Roman' ; trier sur année DESC, puis titre ASC.

**Barème (2 pts).** 1 pt condition WHERE ; 1 pt ORDER BY multicritères.

```
SELECT titre
FROM LIVRE
WHERE genre = 'Roman'
ORDER BY année DESC, titre ASC;
```

## Partie D Agrégat (2 points)

8. Dans la table EMPRUNT, l'attribut date\_retour prend la valeur NULL lorsqu'un livre n'a pas encore été rendu.

Donner une requête SQL calculant le nombre total d'emprunts encore en cours (c.-à-d. date\_retour IS NULL). Nommer la colonne nb\_en\_cours.



### Corrigé

Comptage conditionnel par COUNT (\*) avec IS NULL. Sur l'extrait : 1 emprunt en cours (id\_emprunt=1001).

**Barème (2 pts).** 1 pt COUNT (\*) ; 1 pt condition IS NULL.

```
SELECT COUNT(*) AS nb_en_cours
FROM EMPRUNT
WHERE date_retour IS NULL;
```

## Partie E Requêtes avec jointure (6 points)

9. On souhaite obtenir, pour chaque livre, son titre ainsi que le nom et le prénom de l'auteur, triés par titre croissant. Sur l'extrait, le tableau attendu est :

titre	nom	prénom
1984	<i>Orwell</i>	<i>George</i>
<i>Dune</i>	<i>Herbert</i>	<i>Frank</i>
<i>Ensemble, c'est...</i>	<i>Gavaldà</i>	<i>Anna</i>

Écrire la requête SQL correspondante.



### Corrigé

Jointure LIVRE↔AUTEUR sur id\_auteur; tri par titre.

**Barème (3 pts).** 2 pts jointure correcte (tables, clé, colonnes); 1 pt ORDER BY.

```

SELECT L.titre, A.nom, A.prenom
FROM LIVRE AS L
JOIN AUTEUR AS A ON A.id_auteur = L.id_auteur
ORDER BY L.titre ASC;

```

10. On souhaite obtenir la liste des auteurs (nom, prenom) dont au moins un livre a été emprunté par un adhérent habitant la ville de 'Paris'. Chaque auteur ne doit apparaître qu'une seule fois dans le résultat. Écrire la requête SQL correspondante.



### Corrigé

Chaîne de jointures AUTEUR←LIVRE←EMPRUNT→ADHERENT, filtre Ad.ville='Paris', DISTINCT.

**Barème (3 pts).** 2 pts schéma de jointures correct; 1 pt DISTINCT + condition.

```

SELECT DISTINCT A.nom, A.prenom
FROM AUTEUR AS A
JOIN LIVRE AS L ON L.id_auteur = A.id_auteur
JOIN EMPRUNT AS E ON E.id_livre = L.id_livre
JOIN ADHERENT AS Ad ON Ad.id_adh = E.id_adh
WHERE Ad.ville = 'Paris';

```

### Bonus (optionnel) (2 points, hors notation)

11. Analyse par ville (agrégats).

Pour chaque ville, donner le nombre d'adhérents et le nombre moyen d'emprunts par adhérent. On ne conservera que les villes ayant au moins deux adhérents.



### Corrigé

```

SELECT ADHERENT.ville,
COUNT(DISTINCT ADHERENT.id_adh) nb_adh,
ROUND(COUNT(EMPRUNT.id_emprunt) / COUNT(DISTINCT ADHERENT.id_adh), 2)
emprunts_moy FROM ADHERENT
LEFT JOIN EMPRUNT ON EMPRUNT.id_adh = ADHERENT.id_adh
GROUP BY ADHERENT.ville
HAVING COUNT(DISTINCT ADHERENT.id_adh) >= 2
ORDER BY ADHERENT.ville;

```

↩ Fin du devoir ↪