

- **L3 MIAHS/Ingémath/METIS**
- **Université Paris Cité**
- Année 2024-2025
- [Course Homepage](#)
- [Moodle](#)



- Pas de documents autorisés
- Pas de téléphone portable
- Merci d'écrire vos requêtes en respectant les habitudes d'indentation et de mise en page suivies en cours et dans les documents.

🚲 Un service de vélos partagés permet aux abonnés d'utiliser des vélos mécaniques ou électriques. Chaque vélo porte un numéro. Un vélo entre en service à une date donnée, Il est retiré du service à une date donnée (pas connue à l'avance).

🚲 Chaque abonné souscrit un abonnement pour une durée d'un an à une date donnée. Chaque abonnement possède un numéro, et un titulaire qui possède un nom, un prénom, un âge et un sexe. Un abonnement n'est pas renouvelable. Cela n'empêche pas une personne de souscrire un autre abonnement.

🚲 Un abonné emprunte un vélo à une bornette à un instant de départ donné. Une fois le trajet effectué, l'abonné verrouille le vélo sur une bornette à l'instant d'arrivée.

🚲 Les bornettes sont situées sur des stations. Chaque station contient un nombre n (qui peut varier d'une station à l'autre, mais est constant pour une station donnée) de bornettes numérotées de 1 à n . Les stations sont numérotées, elles ont un nom et une adresse. Chaque station a une longitude et une latitude.

i Question 1 (4pts)

Proposer un diagramme entité-association (EA) correspondant à cette modélisation.
On attend un dessin selon les conventions du cours, pas une énumération.

i Question 2 (2pts)

Lister les contraintes externes

i Question 3 (2pts)

Proposer une traduction en pattes de corbeau du diagramme EA proposé en réponse à la première question.

i Question 3 (1pt)

Proposer un mécanisme pour mettre en place les contraintes externes en SQL lorsque c'est possible sans utiliser les gachettes (**TRIGGER**).

On suppose que le schéma est muni des dépendances fonctionnelles déduites de la question précédente et de celles qui se déduisent des contraintes de clé primaire. On note cet ensemble de dépendances fonctionnelles Σ .

i Question 4 (1pt)

Votre schéma est-il en Forme Normale de Boyce-Codd ?

i Question 5 (2pt)

Quelles actions faut-il effectuer sur votre base pour traduire les événements suivants :

- a. Souscription d'un abonnement.
- b. Mise en service d'un vélo.
- c. Retrait de service d'un vélo.
- d. Décrochage d'un vélo.
- e. Accrochage d'un vélo.

👉 On n'attend pas du code. Juste une phrase qui décrit l'opération à effectuer.

⚠ Attention

Dans la suite, vous formulerez les requêtes dans le schéma relationnel défini par votre schéma en pattes de corbeau.

🎓 : 1 point par requête

i Requête 1

Liste des trajets en cours à un instant donné

i Requête 2

Liste des vélos en trajet depuis plus d'une heure.

i Requête 2

Nombre de trajets initiés durant chaque heure de la journée pendant le mois de juin 2024.

i Requête 3

Liste des vélos qui ont participé à un trajet commencé et achevé dans la même station pendant la dernière semaine.

i Requête 4

Lister d'éventuels couples de trajets suspects impliquant le même vélo à un même instant.

i Requête 5

Liste des stations qui ont été vides ou pleines pendant la semaine écoulée.

i Requête 6

Liste des vélos en service qui n'ont pas roulé depuis plus d'un mois.

i Requête 7


Pour chaque couple de stations, durée moyenne des trajets entre la station de départ et la station d'arrivée.

i Requête 8

Lister pour chaque station le nombre de bornettes occupées à l'instant courant.


Requête 9

Lister les trajets suspects dont le vélo n'est pas en service.


-  En PostgreSQL, pour définir un intervalle à l'aide de deux dates **debut** et **fin**, il suffit d'écrire (**début**, **fin**). L'intervalle ne contient pas la date de fin. Pour tester l'intersection/le recouvrement de deux intervalles, on utilise l'opérateur **OVERLAPS**

```
bd_2023-24=# SELECT
  ('2025-01-03'::date, '2025-01-10'::date) OVERLAPS
  ('2025-01-10'::date, '2025-01-15'::date) ;
overlaps
-----
false
(1 row)


SELECT
  ('2025-01-03 20:26:00'::timestamp, '2025-01-03 21:31:01'::timestamp) OVERLAPS
  ('2025-01-03 20:50:04'::timestamp, '2025-01-03 21:45:00'::timestamp) ;
overlaps
-----
t
(1 row)
```

-  Pour spécifier un intervalle semi-infini (dont la fin n'est pas connue), on peut utiliser **'infinity'::timestamp** pour décrire la borne supérieure.

```
bd_2023-24=# SELECT
  ('2025-01-03 20:26:00'::timestamp, 'infinity'::timestamp) OVERLAPS
  ('2025-01-03 21:32:04'::timestamp, '2025-01-03 21:45:00'::timestamp) ;
overlaps
-----
t
(1 row)
```

-  En PostgreSQL, **current_timestamp** s'évalue à l'instant courant (de type **timestamp** avec **timezone**). En PostgreSQL, pour extraire le mois d'un objet **dd** de type **date**, vous pouvez utiliser **EXTRACT(MONTH FROM dd)**. Le résultat est un entier entre 1 et 12, 1 pour janvier, ...

```
postgres=# SELECT
  current_timestamp AS instant,
  EXTRACT( MONTH FROM current_timestamp::date) AS le_mois ;
          instant              | le_mois
-----+-----
2025-06-05 20:26:12.556256+02 |      6
```

-  Pour définir un intervalle de temps, il suffit de décrire l'intervalle par une chaîne de caractères et de convertir le résultat en type **interval**

```
bd_2023-24=# select '2025-06-05 21:10:38.732237+02'::timestamp - '7 days'::interval ;
?column?
-----
2025-05-29 21:10:38.732237
```