- **L3 MIASHS/Ingémath/METIS**
- **Université Paris Cité**
- Course Homepage

CC1 du 17 octobre 2025

Présentation de la base de données utilisée

Toutes les questions portent sur une base de données `hotel` utilisée par un hôtel pour gérer ses réservations.

Pour chaque relation (table), nous indiquons son schéma avec le type de chaque attribut sous la forme `nom_attribut : type`.

Par défaut, les valeurs `NULL` sont autorisées, sauf si la mention `NOT NULL` est précisée.

Pour chaque table, les valeurs de l'attribut `numéro` sont uniques, autrement dit chaque ligne est identifiée par la valeur de l'attribut `numéro`.

```
client(
    numéro : int (NOT NULL),
    nom : varchar (NOT NULL),
    prénom : varchar (NOT NULL),
    rue : varchar,
    ville : varchar,
    cp : int,
    pays : varchar,
    tel : varchar (NOT NULL),
    email : vachar
)
```

```
chambre(
    numéro : int (NOT NULL),
    étage : int (NOT NULL),
    nblitdouble : int (NOT NULL),
    nblitsimple : int (NOT NULL),
    description : varchar
)
```

- Les attributs `nblitdouble` et `nblitsimple` précisent respectivement le nombre de lits doubles et de lits simples de la chambre.

```
réservation(
    numéro : int (NOT NULL),
    chambre : int (NOT NULL),
    client : int (NOT NULL),
    arrivée : date (NOT NULL),
    départ : date (NOT NULL),
    nbadulte : int (NOT NULL),
    nbenfant : int (NOT NULL)
)
```

- L'attribut `chambre` référence l'attribut `numéro` de la table `chambre`.
- L'attribut `client` référence l'attribut `numéro` de la table `client`.
- Les attributs `nbadulte` et `nbenfant` précisent respectivement le nombre d'adultes et d'enfants qui logeront dans la chambre.
- Le type `date` dispose de la relation d'ordre usuelle.

Voici des exemples de lignes pour chaque table :

```
client
| numéro | nom       | prénom   | rue           | ville     | cp    | pays     |
| 113    | 'Legrand' | 'Claire' | '12 rue d'If' | 'Orléans' | 45000 | 'France' |

 tel          | email                |
 '0610101010' | 'jlegrand@cecher.fr' |
```

```
chambre
| numéro | étage | nblitdouble | nblitsimple | description                                |
| 104    | 1     | 1           | 0           | spacieuse, ensoleillée, balcon, WC, douche |
```

```
réservation
| numéro | chambre | client | arrivée    | départ     | nbadulte | nbenfant |
| 512    | 104     | 113    | 2025/12/15 | 2025/12/21 | 2        | 0        |
```

Cette ligne de la table réservation stocke que le client 113 de la table `client` (Claire Legrand) a réservé la chambre 104 de la table `chambre` du 2025/12/15 au 2025/12/21 pour 2 adultes.

Questions

1. Ecrire **en algèbre relationnelle et en SQL** une requête qui liste les noms et prénoms des clients qui habitent la ville 'Bordeaux'.

Solution

$\Pi(\sigma(\texttt{client},\ \texttt{ville = 'Bordeaux'}),\ \texttt{nom, prénom})$

```sql
SELECT nom, prénom
FROM client
WHERE ville = 'Bordeaux';
```

2. Ecrire **en algèbre relationnelle et en SQL** une requête qui liste les réservations où le nombre de personnes (adultes et enfants) ne respectent pas le nombre de couchages disponibles dans la chambre réservée.

Solution

$\Pi(\sigma(\texttt{réservation} \bowtie_{\texttt{réservation.chambre=chambre.numéro}} \texttt{chambre},$
$\texttt{nbadulte+nbenfant > nblitdouble*2+nblitsimple}), \texttt{réservation.numéro})$

```sql
SELECT r.*
FROM réservation r JOIN chambre ch ON r.chambre = ch.numéro
WHERE r.nbadulte +  r.nbenfant > ch.nblitdouble * 2 + ch.nblitsimple;
```

Une projection `SELECT r.numéro` convient aussi.

3. Ecrire **en SQL** une requête qui liste les numéros des clients qui n'ont aucune réservation au 1er étage.

Solution

Avec une jointure externe et une sous-requête :

```
WITH resaétage1 as
(
    SELECT r.*
    FROM réservation r JOIN chambre ch ON r.chambre = ch.numéro
    WHERE ch.étage = 1
)
SELECT cl.numéro
FROM client cl LEFT JOIN resaétage1 r ON cl.numéro = r.client
WHERE r.client IS NULL;
```

On peut écrire une version avec `IN`, ou bien `EXISTS`, et une sous-requête :

```
SELECT cl.numéro
FROM client cl
WHERE cl.numéro NOT IN(
    SELECT DISTINCT r.client
    FROM réservation r JOIN chambre ch ON r.chambre = ch.numéro
    WHERE ch.étage = 1
    );
```

Une solution équivalente avec `EXCEPT` et deux sous-requêtes :

```
(
    SELECT numéro
    FROM client
)
EXCEPT
(
    SELECT DISTINCT r.client
    FROM réservation r JOIN chambre ch ON r.chambre = ch.numéro
    WHERE ch.étage = 1
);
```

Le distinct limite la complexité de l'opération `EXCEPT`.

4. Ecrire **en SQL** une requête qui liste les noms, prénoms et nombres de réservations des 10 clients qui ont fait le plus de réservations.

Solution

```
SELECT cl.nom, cl.prénom, COUNT(r.numéro) AS nbréservation
FROM client cl JOIN réservation r ON cl.numéro = r.client
GROUP BY cl.numéro, cl.nom, cl.prénom
ORDER BY nbréservation DESC
LIMIT 10;
```

Ici, il faut bien grouper par numéro de client, même si cet attribut n'est pas dans la projection car c'est le seul attribut de la table `client` qui identifie chaque tuple. Le groupement par nom et prénom n'ajoute rien mais est nécessaire pour la projection.

5. Dans cette requête, on recherche les cas où deux réservations différentes pour une même chambre ont au moins une nuit commune. Il s'agit donc de réservations incompatibles.

   Ecrire **en SQL** une requête qui liste, pour chaque incompatibilité, le numéro de la chambre concernée et les numéros des deux réservations.

Solution

Deux réservations sont incompatibles si le client de l'une arrive quand le client de l'autre occupe encore la chambre.

```
SELECT r1.chambre, r1.numéro AS numéroreservation1, r2.numéro AS numéroreservation2
FROM réservation r1
    JOIN réservation r2 ON (r1.chambre = r2.chambre AND r1.numéro <> r2.numéro)
WHERE (r1.arrivée >= r2.arrivée AND r1.arrivée < r2.départ)
    OR (r2.arrivée >= r1.arrivée AND r2.arrivée < r1.départ) ;
```

Avec cette condition, chaque couple de réservations incompatibles apparaît deux fois.
Si on veut qu'ils n'apparaissent qu'une fois :

```
SELECT r1.chambre, r1.numéro AS numéroreservation1, r2.numéro AS numéroreservation2
FROM réservation r1
    JOIN réservation r2 ON (r1.chambre = r2.chambre AND r1.numéro <> r2.numéro)
WHERE r1.arrivée >= r2.arrivée AND r1.arrivée < r2.départ;
```

NB : deux réservations telles que `r1.arrivée = r2.départ` ou `r2.arrivée = r1.départ` ne sont pas incompatibles.

6. Ecrire **en SQL, sans sous-requête et sans INTERSECT**, une requête qui liste les numéros des clients qui ont au moins une réservation pour la chambre 101 **et** au moins une réservation pour la chambre 303.

Solution

```
SELECT DISTINCT r1.client
FROM réservation r1
    JOIN réservation r2 ON r1.client = r2.client
WHERE r1.chambre = 101 AND r2.chambre = 303;
```

NB : ce n'est pas une bonne idée d'utiliser les agrégats pour répondre à un problème du type 'il existe au moins un', 'il existe au moins deux', 'il n'existe pas'.
Si on voulait quand même écrire une requête avec `COUNT`, il fallait grouper par numéro de client et utiliser `CASE WHEN` pour compter en même temps les réservations de la chambre 101 et les réservations de la chambre 303 :

```
SELECT client
FROM réservation
GROUP BY client
HAVING COUNT(
    CASE
      WHEN chambre = 101 THEN numéro
      ELSE NULL
    END) >= 1
    AND COUNT(
    CASE
      WHEN chambre = 303 THEN numéro
      ELSE NULL
    END) >= 1
;
```

Parmi les solutions exclues avec sous-requêtes :

```
WITH client101 AS
(
    SELECT DISTINCT cl.numéro
    FROM client cl JOIN réservation r ON cl.numéro = r.client
    WHERE r.chambre = 101
),
client303 AS
(
    SELECT DISTINCT cl.numéro
    FROM client cl JOIN réservation r ON cl.numéro = r.client
    WHERE r.chambre = 303
)
SELECT *
FROM client101 NATURAL JOIN client303;
```

On peut écrire deux autres versions semblables avec `INTERSECT` ou `IN`.

7. Ecrire **en SQL, sans fonction d'agrégation et sans sous-requête**, une requête qui liste les numéros des chambres qui ont été réservées par au moins deux personnes différentes de la ville de 'Brest'.

Solution

```
SELECT DISTINCT ch.numéro
FROM client cl1
    JOIN réservation r ON cl1.numéro = r.client
    JOIN client cl2 ON r.client = cl2.numéro
WHERE cl1.ville = 'Brest' AND cl2.ville = 'Brest' AND cl1.numéro <> cl2.numéro;
```

8. Ecrire **en SQL sans fonction d'agrégation**, une requête qui liste les étages pour lesquels toutes les chambres sont réservées à la date '2025/12/15'.

Solution

Il faut faire la différence entre l'ensemble des étages et l'ensemble des étages ayant au moins une chambre disponible le '2025/12/15'.
Avec des sous-requêtes, une jointure externe et une différence :

```
WITH resa15-12-2025 AS
(   -- réservations pour le 2025/12/15
    SELECT *
    FROM réservation
    WHERE arrivée <= '2025/12/15' AND départ > '2025/12/15'
)
(
    SELECT DISTINCT étage
    FROM chambre
)
EXCEPT
(   -- étages avec au moins une chambre dispo le 2025/12/15
    SELECT DISTINCT ch.étage
    FROM chambre ch LEFT JOIN resa15-12-2025 r ON ch.numéro = r.chambre
    WHERE r.chambre IS NULL
);
```

Avec des sous-requêtes, `NOT IN` et une différence :

```
(
    SELECT DISTINCT étage
    FROM chambre
)
EXCEPT
(
    SELECT DISTINCT étage -- étages avec au moins une chambre dispo le 2025/12/15
    FROM chambre
    WHERE numéro NOT IN
        (   -- chambres réservées le 2025/12/15
            SELECT chambre
            FROM réservation
            WHERE arrivée <= '2025/12/15' AND départ > '2025/12/15'
        )
);
```