2024 /2025

# Les requêtes en algèbre relationnelle

**FACULTE DES SCIENCES SEMLALIA - MARRAKECH**

**MODULE** : BASES DE DONNES RELATIONNELLE

**FILIERE** : IAP-S4

**ENCADRE PAR** : PROF. J.ZAHIR

**REALISE PAR** :

MAAROUF QAMAR

EL-MAHJOUB BOUNHAR

2332259 - 2330686

# Introduction

Ce document présente les expressions d'algèbre relationnelle correspondant aux requêtes SQL du projet de gestion hôtelière.

### Notations utilisées :

- σ : Sélection (WHERE)
- π : Projection (SELECT)
- ⋈ : Jointure naturelle (JOIN)
- ⋈c : Jointure avec condition
- ∪ : Union
- ∩ : Intersection
- Différence
- ρ : Renommage
- γ : Groupement (GROUP BY)

---

## a. Liste des réservations avec nom du client et ville de l'hôtel

### Requête SQL :

sql

```
SELECT R.Id_Reservation, C.Nom_complet, H.Ville AS Ville_Hotel
FROM Reservation R
JOIN Client C ON R.Id_Client = C.Id_Client
JOIN Chambre Ch ON R.Id_Reservation = Ch.Id_Chambre
JOIN Hotel H ON Ch.Id_Hotel = H.Id_Hotel;
```

### Expression d'algèbre relationnelle :

π Id_Reservation, Nom_complet, ρ(Ville_Hotel ← Ville)(H.Ville) (

Reservation ⋈ Id_Client=Id_Client Client ⋈ Id_Reservation=Id_Chambre Chambre ⋈ Id_Hotel=Id_Hotel Hotel)

### Décomposition étape par étape :

1. **Temp1** = Reservation ⋈ Id_Client=Id_Client Client
2. **Temp2** = Temp1 ⋈ Id_Reservation=Id_Chambre Chambre
3. **Temp3** = Temp2 ⋈ Id_Hotel=Id_Hotel Hotel
4. **Résultat** = π Id_Reservation, Nom_complet, ρ(Ville_Hotel ← Ville)(Ville) (Temp3)

---

## b. Clients qui habitent à Paris

### Requête SQL :

sql

```
SELECT *
FROM Client
WHERE Ville = 'Paris';
```

### Expression d'algèbre relationnelle :

σ Ville='Paris' (Client)

---

## c. Nombre de réservations par client

### Requête SQL :

sql

```
SELECT C.Nom_complet, COUNT(R.Id_Reservation) AS Nombre_Reservations
FROM Client C
LEFT JOIN Reservation R ON C.Id_Client = R.Id_Client
GROUP BY C.Id_Client;
```

### Expression d'algèbre relationnelle :

π Nom_complet, ρ(Nombre_Reservations ← COUNT(Id_Reservation)) (

γ Id_Client; COUNT(Id_Reservation) (Client ⟕ Id_Client=Id_Client Reservation)

)

### Notation :

- ⟕ représente la jointure externe gauche (LEFT JOIN)
- γ A; f(B) représente le groupement par A avec fonction d'agrégation f(B)

---

## d. Nombre de chambres par type

### Requête SQL :

sql

```
SELECT T.Type, COUNT(Ch.Id_Chambre) AS Nombre_Chambres
FROM Type_Chambre T
LEFT JOIN Chambre Ch ON T.Id_Type = Ch.Id_Type
GROUP BY T.Type;
```

### Expression d'algèbre relationnelle :

π Type, ρ(Nombre_Chambres ← COUNT(Id_Chambre)) (

γ Type; COUNT(Id_Chambre) (Type_Chambre ⟕ Id_Type=Id_Type Chambre)

)

---

## e. Chambres non réservées pour une période donnée

### Requête SQL :

sql

```
SET @date_debut = '2025-07-01';
SET @date_fin = '2025-07-10';
SELECT *
FROM Chambre Ch
```

```
WHERE Ch.Id_Chambre NOT IN (
    SELECT Co.Id_Chambre
    FROM Concerner Co
    JOIN Reservation R ON Co.Id_Reservation = R.Id_Reservation
    WHERE R.Date_arrivee <= @date_fin AND R.Date_depart >= @date_debut
);
```

## Expression d'algèbre relationnelle :

### Sous-requête (chambres réservées) :

$$\pi_{Id\_Chambre}\left(\sigma_{Date\_arrivee \leq '2025\text{-}07\text{-}10' \wedge Date\_depart \geq '2025\text{-}07\text{-}01'}\left(Concerner \bowtie_{Id\_Reservation=Id\_Reservation} Reservation\right)\right)$$

### Requête principale :

$$Chambre - \left(Chambre \bowtie_{Id\_Chambre \in \{chambres\_reservees\}} \left(\pi_{Id\_Chambre}\left(\sigma_{Date\_arrivee \leq '2025\text{-}07\text{-}10' \wedge Date\_depart \geq '2025\text{-}07\text{-}01'}\left(Concerner \bowtie_{Id\_Reservation=Id\_Reservation} Reservation\right)\right)\right)\right)$$

## Ou plus simplement avec l'anti-jointure :

$$Chambre \bowtie \left(\pi_{Id\_Chambre}\right.$$

σ Date_arrivee≤'2025-07-10' ∧ Date_depart≥'2025-07-01' (

Concerner ⋈ Id_Reservation=Id_Reservation Reservation

)

)

)

**Notation :** ⋉ représente l'anti-jointure (NOT IN)

---

# Résumé des opérateurs utilisés

| Opérateur | Signification | Équivalent SQL |
|---|---|---|
| σ | Sélection | WHERE |
| π | Projection | SELECT |
| ⋈ | Jointure | JOIN |
| ⟕ | Jointure externe gauche | LEFT JOIN |
| ⋉ | Anti-jointure | NOT IN |
| γ | Groupement | GROUP BY |
| ρ | Renommage | AS |
| - | Différence | EXCEPT |

## Notes importantes

- Les expressions peuvent être optimisées selon l'ordre des opérations
- La jointure naturelle (⋈) assume l'égalité sur les attributs de même nom
- Les fonctions d'agrégation (COUNT, SUM, etc.) sont notées dans le groupement γ

# Question 4 : SQLite vs MySQL

## Qu'est-ce que SQLite et quelle différence avec MySQL ?

### SQLite

**SQLite est un système de gestion de base de données relationnelle (SGBDR) léger, autonome et sans serveur. Il stocke toute la base de données dans un seul fichier sur le disque.**

## Principales différences entre SQLite et MySQL :

### Architecture

- **SQLite : Base de données embarquée, sans serveur, stockée dans un fichier local**
- **MySQL : Base de données client-serveur nécessitant un serveur dédié**

### Installation et configuration

- **SQLite : Aucune installation serveur requise, bibliothèque intégrée dans l'application**
- **MySQL : Installation et configuration d'un serveur MySQL nécessaires**

### Concurrence

- **SQLite : Support limité de la concurrence (un seul écrivain à la fois)**
- **MySQL : Excellente gestion de la concurrence avec de nombreux utilisateurs simultanés**

### Performance

- **SQLite : Très rapide pour les petites à moyennes bases de données et applications mono-utilisateur**
- **MySQL : Optimisé pour les gros volumes de données et les applications multi-utilisateurs**

### Cas d'usage

- **SQLite : Applications mobiles, prototypes, développement local, applications embarquées**

- **MySQL : Applications web, sites e-commerce, systèmes d'entreprise, applications nécessitant de nombreux utilisateurs simultanés**

## Avantages de SQLite

- **Simplicité d'utilisation**
- **Pas de configuration requise**
- **Portable (un seul fichier)**
- **Idéal pour le développement et les tests**

## Avantages de MySQL

- **Meilleure scalabilité**
- **Gestion avancée des utilisateurs et permissions**
- **Réplication et clustering**
- **Outils d'administration riches**

**Pour ce projet, SQLite est parfait car il s'agit d'une application de démonstration avec une interface Streamlit, ne nécessitant pas la complexité d'un serveur MySQL**