1. SQL-89 et SQL-92 : Différences de Syntaxe de Jointures

- **SQL-89** : Les jointures sont écrites en listant les tables dans la clause `FROM`, puis en utilisant des conditions dans `WHERE` pour relier les tables.
    - **Exemple de jointure** :

      ```
      sql

      SELECT c.numCh, c.catCh, r.numCli, r.dateDeb, r.nbJours
      FROM Chambre c, Réservation r
      WHERE c.numHot = r.numHot
        AND c.numCh = r.numCh;
      ```

    - **Limitation** : SQL-89 ne supporte pas les jointures externes (comme `LEFT JOIN`) de manière standard. Cependant, certains SGBD comme Oracle utilisent une syntaxe spécifique avec `(+)` pour indiquer une jointure externe :

      ```
      sql

      FROM Chambre c, Réservation r
      WHERE c.numHot = r.numHot(+)
        AND c.numCh = r.numCh(+);
      ```

- **SQL-92** : Introduit les `JOIN` explicites, rendant les requêtes plus lisibles et permettant les **jointures externes** (`LEFT JOIN`, `RIGHT JOIN`, `FULL JOIN`).
    - **Exemple de jointure interne** :

      ```
      sql

      SELECT c.numCh, c.catCh, r.numCli, r.dateDeb, r.nbJours
      FROM Chambre c
      JOIN Réservation r ON c.numHot = r.numHot AND c.numCh = r.numCh;
      ```

    - **Exemple de jointure externe gauche** (pour lister toutes les chambres d'un hôtel même si elles n'ont pas de réservation) :

      ```
      sql

      SELECT c.numCh, c.catCh, r.numCli, r.dateDeb, r.nbJours
      FROM Chambre c
      LEFT JOIN Réservation r ON c.numHot = r.numHot AND c.numCh = r.numCh
      WHERE c.numHot = 74;
      ```

2. Types de Données Importants

- **DATE** : Type de données pour les colonnes qui stockent des dates. On utilise souvent le format `'YYYY-MM-DD'` lors des insertions.
    - **Exemple** :

      ```
      sql

      CREATE TABLE Réservation (
          numRes INT PRIMARY KEY,
          dateDeb DATE,
          nbJours INT
      );
      ```

3. Clés Primaires et Étrangères

- **Clé primaire (PRIMARY KEY)** : Garantit l'unicité de chaque ligne dans la table.
    - **Exemple** :

      ```sql
      CREATE TABLE Client (
          numCli INT PRIMARY KEY,
          nomCli VARCHAR(100)
      );
      ```

- **Clé étrangère (FOREIGN KEY)** : Lie une colonne de la table actuelle à la clé primaire d'une autre table pour garantir l'intégrité référentielle.
    - **Exemple** :

      ```sql
      CREATE TABLE Réservation (
          numRes INT PRIMARY KEY,
          numCli INT,
          FOREIGN KEY (numCli) REFERENCES Client(numCli)
      );
      ```

    - **Options supplémentaires** :
        - **ON DELETE CASCADE** : Supprime les enregistrements dans la table secondaire si la ligne correspondante dans la table primaire est supprimée.
        - **ON UPDATE CASCADE** : Met à jour automatiquement les valeurs de la clé étrangère dans la table secondaire si elles changent dans la table primaire.

4. Conditions et Clauses Essentielles

- **WHERE** : Filtre les résultats en fonction d'une condition.
    - **Exemple** :

      ```sql
      SELECT * FROM Client WHERE vilCli = 'Paris';
      ```

- **NOT IN** : Utilisé pour sélectionner les enregistrements qui ne figurent pas dans une sous-requête.
    - **Exemple** (Liste des clients sans réservation le 1er décembre 2023) :

      ```sql
      SELECT numCli, nomCli
      FROM Client
      WHERE numCli NOT IN (
          SELECT numCli
          FROM Réservation
          WHERE dateDeb <= '2023-12-01'
            AND dateDeb + INTERVAL nbJours DAY > '2023-12-01'
      );
      ```

5. Gestion des Dates et Périodes

- **Comparaison des Dates** : On peut comparer des dates avec `=`, `<`, `>`, etc.
- **Addition de jours à une date** : En SQL-92 et dans certains SGBD, on peut ajouter un intervalle à une date avec `INTERVAL`.
    - **Exemple** :

      ```
      sql

      WHERE dateDeb <= '2023-12-01'
        AND dateDeb + INTERVAL nbJours DAY > '2023-12-01'
      ```

6. Requêtes de Manipulation des Données

- **INSERT INTO** : Insère un nouveau tuple (ou enregistrement) dans une table.
    - **Exemple** :

      ```
      sql

      INSERT INTO Client (numCli, nomCli, adrCli, vilCli)
      VALUES (77, 'Dupond', 'Rue de la Promenade', 'Nice');
      ```

- **UPDATE** : Modifie les valeurs d'un ou plusieurs enregistrements dans une table.
    - **Exemple** :

      ```
      sql

      UPDATE Hôtel
      SET catHot = 4
      WHERE numHot = 10;
      ```

- **DELETE FROM** : Supprime un ou plusieurs enregistrements dans une table.
    - **Exemple** :

      ```
      sql

      DELETE FROM Réservation
      WHERE numCli = 48;
      ```

Conseils et Tips pour SQL

- **Bien identifier les colonnes `NOT NULL`** : Lors de l'insertion, toutes les colonnes marquées `NOT NULL` nécessitent une valeur.
- **Utiliser des alias (`AS`) pour simplifier les requêtes** : Par exemple, `FROM Chambre AS C` permet de raccourcir l'écriture.
- **Préférer `JOIN` explicites en SQL-92** : Les `JOIN` rendent la requête plus lisible et plus facile à déboguer.
- **Attention aux requêtes sans `WHERE` avec `DELETE` et `UPDATE`** : Sans `WHERE`, toutes les lignes seront affectées.

1. Maîtrisez les Types de Jointures

- **Jointure Interne (INNER JOIN)** : Récupère uniquement les lignes qui ont une correspondance dans les deux tables. Utilisez-la pour les requêtes où les données sont assurées d'exister des deux côtés.
    - **Exemple** : Rechercher les clients qui ont fait une réservation :

      ```
      sql
      Copier le code
      SELECT Client.numCli, Client.nomCli, Réservation.dateDeb
      FROM Client
      JOIN Réservation ON Client.numCli = Réservation.numCli;
      ```

- **Jointure Externe Gauche (LEFT JOIN)** : Récupère toutes les lignes de la table de gauche, même si elles n’ont pas de correspondance dans la table de droite. Utilisez-la pour inclure des lignes qui n’ont pas encore de données associées.
    - **Exemple** : Lister toutes les chambres d'un hôtel, y compris celles sans réservation :

      ```
      sql
      Copier le code
      SELECT Chambre.numCh, Réservation.dateDeb
      FROM Chambre
      LEFT JOIN Réservation ON Chambre.numHot = Réservation.numHot AND
      Chambre.numCh = Réservation.numCh;
      ```

2. Utilisez les Sous-requêtes Judicieusement

- **Sous-requête avec `NOT IN`** : Pour sélectionner des éléments non présents dans une autre table (par exemple, les clients sans réservation un jour donné), les sous-requêtes sont idéales.
- **Sous-requête dans `WHERE`** : Très utile pour filtrer des données en fonction de résultats dynamiques.
    - **Exemple** : Lister les clients sans réservation le 1er décembre :

      ```
      sql
      Copier le code
      SELECT numCli, nomCli
      FROM Client
      WHERE numCli NOT IN (
          SELECT numCli
          FROM Réservation
          WHERE dateDeb <= '2023-12-01' AND dateDeb + INTERVAL nbJours DAY
      > '2023-12-01'
      );
      ```

3. Manipulation des Dates

- **Calculs de Date** : Utilisez `INTERVAL` pour ajouter ou soustraire des jours, mois, années.
    - **Exemple** : Calculer une date de fin de réservation :

      ```
      sql
      Copier le code
      SELECT dateDeb + INTERVAL nbJours DAY AS dateFin
      FROM Réservation;
      ```

- **Comparaison de Dates** : Les conditions `<=`, `>=` et `BETWEEN` permettent de filtrer des périodes.
    - **Exemple** : Sélectionner les réservations en cours le 1er décembre :

      ```
      sql
      Copier le code
      WHERE dateDeb <= '2023-12-01' AND dateDeb + nbJours > '2023-12-01';
      ```

4. Attention aux Clauses `WHERE` pour `DELETE` et `UPDATE`

- **Supprimer avec `DELETE`** : Toujours spécifier une clause `WHERE` pour éviter de supprimer toutes les lignes par erreur.
    - **Exemple** :

      ```
      sql
      Copier le code
      DELETE FROM Réservation WHERE numCli = 48;
      ```

- **Mettre à jour avec `UPDATE`** : Spécifiez des critères pour ne modifier que les lignes voulues.
    - **Exemple** :

      ```
      sql
      Copier le code
      UPDATE Hôtel SET catHot = 4 WHERE numHot = 10;
      ```

5. Vérifiez les Contraintes de Clés Étrangères et Primaires

- **Clés Primaires (PRIMARY KEY)** : Assurent l'unicité et servent souvent pour les jointures.
- **Clés Étrangères (FOREIGN KEY)** : Elles lient les tables. En supprimant ou en modifiant des données, faites attention aux dépendances référentielles (utilisez `ON DELETE CASCADE` si vous voulez que les données liées soient supprimées automatiquement).
    - **Exemple** :

      ```
      sql
      Copier le code
      CREATE TABLE Réservation (
          numRes INT PRIMARY KEY,
          numCli INT,
          FOREIGN KEY (numCli) REFERENCES Client(numCli) ON DELETE CASCADE
      );
      ```

6. Organisez Vos Requêtes Complexes

- Utilisez des **alias** pour simplifier la lecture (ex., `JOIN Client c ON ...`).
- **Indentez vos requêtes** pour une lisibilité accrue, surtout avec des jointures et sous-requêtes multiples.

1. Fonctions d’Agrégation

Les fonctions d’agrégation sont particulièrement utiles pour effectuer des calculs sur un ensemble de lignes.

- **`COUNT()`** : Compte le nombre de lignes. Utile pour connaître le nombre d'enregistrements correspondant à une condition.

  ```
  sql
  Copier le code
  SELECT COUNT(*) FROM Client;
  ```

- **`SUM()`** : Additionne les valeurs d’une colonne numérique.

  ```
  sql
  Copier le code
  SELECT SUM(nbJours) AS totalJours FROM Réservation;
  ```

- **`AVG()`** : Calcule la moyenne des valeurs d’une colonne numérique.

  ```
  sql
  Copier le code
  SELECT AVG(nbJours) AS dureeMoyenne FROM Réservation;
  ```

- **`MIN()` et `MAX()`** : Renvoie la valeur minimale ou maximale d’une colonne.

  ```
  sql
  Copier le code
  SELECT MIN(dateDeb) AS datePremiereReservation FROM Réservation;
  ```

---

2. Clauses Utiles pour le Groupement et Filtrage

- **`GROUP BY`** : Utilisé avec les fonctions d’agrégation pour regrouper les données par une ou plusieurs colonnes.

  ```
  sql
  Copier le code
  SELECT numHot, COUNT(*) AS nbReservations
  FROM Réservation
  GROUP BY numHot;
  ```

- **`HAVING`** : Filtre les résultats après le groupement (`GROUP BY`), permettant de filtrer sur les résultats d’agrégation.

  ```
  sql
  Copier le code
  SELECT numHot, COUNT(*) AS nbReservations
  FROM Réservation
  GROUP BY numHot
  HAVING nbReservations > 5;
  ```

---

3. Fonctions de Chaîne de Caractères

Les fonctions de manipulation de chaînes sont utiles pour formater et extraire des informations textuelles.

- **`CONCAT()`** : Combine deux chaînes. En SQL-92, l'opérateur `||` est aussi utilisé pour la concaténation.

```
sql
Copier le code
SELECT CONCAT(nomCli, ' ', adrCli) AS nomAdresse FROM Client;
```

- **`UPPER()` et `LOWER()`** : Convertit les chaînes de caractères en majuscules ou minuscules.

```
sql
Copier le code
SELECT UPPER(nomCli) FROM Client;
```

- **`SUBSTRING()`** : Extrait une partie d'une chaîne.

```
sql
Copier le code
SELECT SUBSTRING(nomCli, 1, 3) AS abreviationNom FROM Client;
```

---

4. Fonctions de Date

Manipuler les dates est important pour les opérations de planification ou les calculs de durée.

- **`DATEADD()`** (variante selon SGBD) : Ajoute un intervalle à une date. Certaines bases utilisent `INTERVAL` avec `+` pour ajouter des jours ou mois à une date.

```
sql
Copier le code
SELECT dateDeb + INTERVAL nbJours DAY AS dateFin FROM Réservation;
```

- **`DATEDIFF()`** (disponible dans certains SGBD) : Calcule la différence entre deux dates.

```
sql
Copier le code
SELECT DATEDIFF('2023-12-01', dateDeb) AS joursEcoule FROM Réservation;
```

---

5. Clauses de Tri et Limitation de Résultats

- **`ORDER BY`** : Trie les résultats par une ou plusieurs colonnes. Peut être utilisé avec `ASC` pour ordre croissant et `DESC` pour ordre décroissant.

```
sql
Copier le code
SELECT * FROM Client ORDER BY nomCli ASC;
```

- **`LIMIT`** (ou `TOP` dans certains SGBD) : Limite le nombre de lignes retournées.

```
sql
Copier le code
SELECT * FROM Client LIMIT 10;
```

---

6. CASE pour des Conditions dans les Sélections

La clause `CASE` permet d'ajouter des conditions directement dans les requêtes pour afficher différents résultats selon les valeurs.

```
sql
Copier le code
SELECT nomCli,
       CASE
           WHEN vilCli = 'Paris' THEN 'Client Parisien'
           ELSE 'Autre Ville'
       END AS TypeClient
FROM Client;
```

- **Explication** : `CASE` affiche `"Client Parisien"` si `vilCli` est `"Paris"`, sinon `"Autre Ville"`.

---

7. Fonction de Coalescence (`COALESCE`)

`COALESCE()` renvoie la première valeur non nulle dans une liste d'expressions, très utile pour gérer les valeurs `NULL`.

```
sql
Copier le code
SELECT nomCli, COALESCE(adrCli, 'Adresse inconnue') AS adresse FROM Client;
```

- **Explication** : Si `adrCli` est `NULL`, la valeur `"Adresse inconnue"` sera affichée à la place.

---

8. Création et Gestion des Index (pour Optimisation)

Les index améliorent la vitesse de recherche sur des colonnes fréquemment utilisées dans les `WHERE` ou `JOIN`.

```
sql
Copier le code
CREATE INDEX idx_client_nom ON Client(nomCli);
```

- **Explication** : Cet index accélère les recherches basées sur `nomCli`.