Chapitre 3
La manipulation des données (LMD)

1. Introduction

Le langage de manipulation de données permet aux utilisateurs et aux développeurs d'accéder aux données de la base, de modifier leur contenu, d'insérer ou de supprimer des lignes.

Il s'appuie sur quatre ordres de base qui sont SELECT, INSERT, DELETE et UPDATE.

Ces quatre ordres ne sont pas toujours autorisés par l'administrateur de la base qui est le seul à pouvoir attribuer ou non les droits d'utilisation sur ces ordres.

Pour l'utilisateur lambda, il pourra indiquer que seul l'ordre SELECT est utilisable. Les ordres de modification de la base ne sont pas accessibles pour certains utilisateurs pour des raisons évidentes de sécurité.

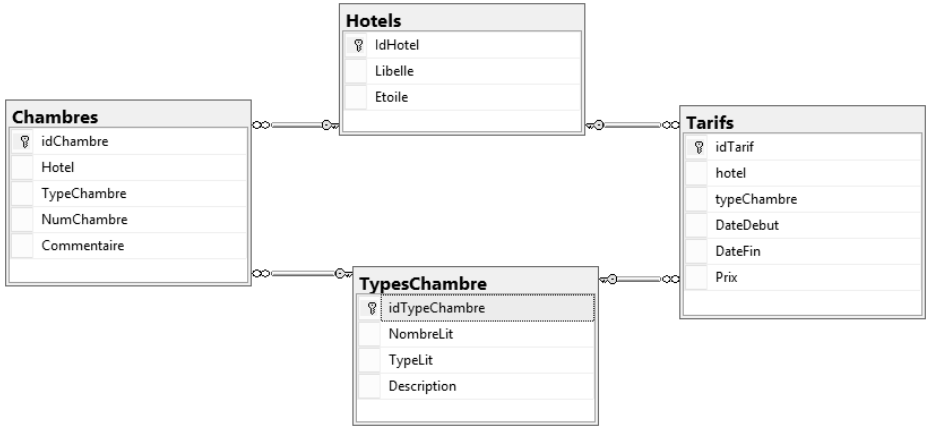
2. La sélection de données

L'ordre SELECT permet de réaliser des requêtes simples assez rapidement même sans connaissances approfondies en langage de programmation. C'est l'ordre de base qui permet d'indiquer au serveur que l'on désire extraire des données.

Il peut également être très puissant si l'on connaît toutes les fonctions et toutes les possibilités du langage. On peut réaliser des requêtes complexes, avec de nombreuses tables mais il faut toujours faire attention aux performances qui peuvent se dégrader très rapidement sur un ordre SQL mal construit ou n'utilisant pas les bons index dans les tables. Il faut être vigilant et utiliser les outils d'analyse de requête (cf. chapitre Approfondissement - Quelques notions de performances) avant d'exécuter une requête sur une base réelle avec des tables conséquentes.

Les principaux éléments d'une requête de sélection	
Clause	**Expression**
SELECT	Liste des colonne(s) et/ou des éléments d'extraction
FROM	Table(s) source(s)
WHERE	Condition(s) ou restriction(s), optionnelle
GROUP BY	Regroupement(s), optionnelle
HAVING	Condition(s) ou restriction(s) sur le(s) regroupement(s), optionnelle
ORDER BY	Tri(s)

Les tables de base qui sont utilisées dans les sections suivantes sont celles-ci :



Attention, les résultats complets des requêtes ne sont pas toujours présentés. Parfois, seules les premières lignes sont affichées.

2.1 L'ordre de sélection de données SELECT

Le SELECT est l'ordre le plus important et le plus utilisé en SQL. Avec cet ordre, nous pouvons ramener des lignes d'une ou plusieurs tables mais également transformer des données par l'utilisation de fonction ou encore réaliser des calculs.

Nous allons décrire progressivement les possibilités de cet ordre dans les paragraphes suivants.

L'utilisation la plus courante consiste à sélectionner des lignes dans une table comme ceci :

```
SELECT NombreLit, Description FROM TypesChambre;
```

Dans cet exemple, nous avons sélectionné deux colonnes de la table TypesChambre.

L'ordre va donc nous ramener toutes les lignes de la table pour ces deux colonnes.

NombreLit	Description
1	1 lit simple avec douche
2	2 lits simples avec douche
2	2 lits simples avec douche et WC séparés
1	1 lit double avec douche
1	1 lit double avec douche et WC séparés
1	1 lit double avec bain et WC séparés
1	1 lit double large avec bain et WC séparés

Si on avait voulu toutes les colonnes et toutes les lignes de la table, l'ordre aurait été celui-ci :

```
SELECT * FROM TypesChambre;
```

idTypeChambre	NombreLit	TypeLit	Description
1	1	lit simple	1 lit simple avec douche
2	2	lit simple	2 lits simples avec douche
3	2	lit simple	2 lits simples avec douche et WC séparés
4	1	lit double	1 lit double avec douche
5	1	lit double	1 lit double avec douche et WC séparés
6	1	lit double	1 lit double avec bain et WC séparés
7	1	lit XL	1 lit double large avec bain et WC séparés

L'étoile est pratique lorsque l'on ne connaît pas le nom des colonnes, mais le résultat est rarement lisible avec des tables contenant un nombre important de colonnes. Pour connaître le nom des colonnes, faites un DESC de la table auparavant (DESC <nom table>).

La syntaxe simple est donc :

```
SELECT <colonne 1>, <colonne 2> ... | * FROM <table1>, <table2> ...
```

Si certaines colonnes ont le même nom mais appartiennent à des tables différentes, il faudra ajouter le nom de la table devant la colonne afin que le système sache quelle colonne prendre.

Il n'est pas obligatoire de mettre le nom des tables devant chaque colonne, mais pour une question de lisibilité et de maintenance, il est préférable de les mettre sur des sélections complexes.

```
SELECT Hotels.Libelle
, Hotels.Etoile
, Chambres.NumChambre
, TypesChambre.Description
FROM Chambres, Hotels, TypesChambre;
```

Nous verrons dans la section L'utilisation des alias de ce chapitre que pour alléger la lecture il est également possible de donner un alias à chaque table. Cet alias est souvent simple et permet de retrouver facilement la table concernée, par exemple CH pour Chambres ou TYP pour TypesChambre.

2.2 Les options DISTINCT et ALL

Par défaut, lors de l'exécution d'un SELECT, toutes les lignes sont ramenées (l'option ALL est automatique). Si l'on veut supprimer les doublons, il faut ajouter l'ordre DISTINCT.

L'ordre DISTINCT s'applique à toutes les colonnes présentes.

Exemple

```
SELECT NombreLit, TypeLit, Description FROM TypesChambre;
```

et :

```
SELECT DISTINCT NombreLit, TypeLit, Description FROM TypesChambre;
```

Les deux SELECT ci-dessus ont le même résultat car il y a un doublon sur les trois premières lignes. Ces deux premières colonnes sont identiques mais pas la troisième.

NombreLit	TypeLit	Description
1	**lit double**	1 lit double avec bain et WC séparés
1	**lit double**	1 lit double avec douche
1	**lit double**	1 lit double avec douche et WC séparés
1	lit simple	1 lit simple avec douche
1	lit XL	1 lit double large avec bain et WC séparés
...		

En revanche, si on réduit la sélection à deux colonnes comme :

```
SELECT NombreLit, TypeLit FROM TypesChambre;
```

on obtient :

NombreLit	TypeLit
1	lit simple
2	lit simple

NombreLit	TypeLit
2	lit simple
1	lit double
1	lit double
...	

Si on ajoute un ordre DISTINCT, une des deux lignes contenant '2' et 'lit simple' sera supprimée.

```
SELECT DISTINCT NombreLit, TypeLit FROM TypesChambre;
```

Remarque

La clause DISTINCT ne peut pas être utilisée avec des opérateurs de regroupement (voir le GROUP BY). En effet, les opérateurs de type COUNT ou SUM éliminent automatiquement les doublons.

NombreLit	TypeLit
1	lit double
1	lit simple
1	lit XL
2	lit simple

2.3 Les tris

Lorsque l'on ramène des colonnes d'une ou de plusieurs tables avec un SELECT, il est souvent intéressant d'obtenir un résultat trié sur certaines colonnes.

Pour cela, on utilisera la clause ORDER BY en fin de requête. On peut trier sur n'importe quelles colonnes d'une table, il faut pour cela que les colonnes fassent partie de la sélection, mais elles ne sont pas obligatoirement affichées.

La clause ne peut être utilisée qu'une seule fois dans une requête et doit toujours être la dernière clause de la requête.

Le tri par défaut est ascendant, noté ASC (du plus petit au plus grand). Il est possible d'indiquer que l'on désire réaliser le tri en descendant en notant DESC.

Syntaxe de l'ORDER BY

```
ORDER BY <colonne 1> [ASC|DESC], <colonne 2> [ASC|DESC]...
```

Exemple

Tri sur la DateDebut de la table Tarifs en ascendant et tri sur le prix de la table en descendant.

```
SELECT hotel
, typeChambre
, DateDebut
, prix
FROM Tarifs
ORDER BY DateDebut, prix DESC;
```

Hotel	typeChambre	DateDebut	prix
1	7	01/04/2026	139,90
4	7	01/04/2026	134,90
1	6	01/04/2026	129,90
4	6	01/04/2026	124,90
1	5	01/04/2026	119,90
1	3	01/04/2026	119,90

On constate que le tri se fait sur la colonne prix du plus grand au plus petit (DESC).

Il existe également la possibilité d'indiquer l'ordre de la colonne dans le SELECT à la place de son nom, à condition que la colonne soit présente dans la sélection.

Exemple sans le nom des colonnes

```
SELECT hotel
, typeChambre
, DateDebut
, prix
FROM Tarifs
ORDER BY 3, 4 DESC;
```

Certes, cette notation fonctionne et permet de ne pas ressaisir le nom des colonnes, mais n'aide vraiment pas à la lisibilité de la requête. Avec des sélections multiples, l'ordre devient vite illisible pour celui qui ne l'a pas écrit.

Remarque

Attention : le tri peut être source de baisse de performance de la requête. En effet, si la requête ramène des millions de lignes, le système va récupérer toutes les lignes dans un premier temps, puis trier l'ensemble et ensuite seulement commencer à restituer les éléments dans l'ordre. Le système va utiliser de l'espace disque pour stocker les éléments à trier puis de la mémoire pour réaliser le tri.

Le tri est déconseillé dans le cas où il n'y a pas de clause WHERE ou que celle-ci est peu restrictive et que la table contient des millions d'enregistrements.

Remarque

Autre remarque, les performances seront meilleures en réalisant les tris sur des colonnes indexées.

2.4 Les options TOP, LIMIT, OFFSET ou ROWNUM

L'exécution d'un SELECT ramène toutes les lignes. Si l'on veut obtenir un nombre de lignes maximum dans le résultat, comme les 10 premières lignes, il faut utiliser l'ordre TOP ou LIMIT selon le SGDBR utilisé. Si l'on souhaite les dernières lignes, il est nécessaire d'ajouter un tri décroissant. L'ordre OFFSET ajouté à l'ordre LIMIT permet d'effectuer un décalage de résultat. L'ordre ROWNUM correspond à la numérotation des lignes d'une requête et permet donc aussi d'afficher une partie du résultat selon le besoin.

Syntaxe SQL Server pour les 10 premières puis les 10 dernières lignes

```
SELECT TOP 10 idTarif, hotel, typechambre, dateDebut, DateFin,
Prix
FROM Tarifs;

SELECT TOP 10 idTarif, hotel, typechambre, dateDebut, DateFin,
Prix
FROM Tarifs ORDER BY idTarif DESC;
```

Syntaxe MySQL et PostgreSQL

Les 10 premières lignes :

```
SELECT idTarif, hotel, typechambre, dateDebut, DateFin, Prix FROM
Tarifs LIMIT 10;
```

Les 10 premières lignes à partir de la cinquième ligne :

```
SELECT idTarif, hotel, typechambre, dateDebut, DateFin, Prix FROM
Tarifs LIMIT 10 OFFSET 5;
```

ou :

```
SELECT idTarif, hotel, typechambre, dateDebut, DateFin, Prix FROM
Tarifs LIMIT 5, 10;
```

Les 10 dernières lignes :

```
SELECT idTarif, hotel, typechambre, dateDebut, DateFin, Prix FROM
Tarifs ORDER BY idTarif DESC LIMIT 10;
```

Syntaxe Oracle

```
SELECT idTarif, hotel, typechambre, dateDebut, DateFin, Prix FROM
Tarifs
WHERE ROWNUM <= 10
ORDER BY ROWNUM DESC;
```