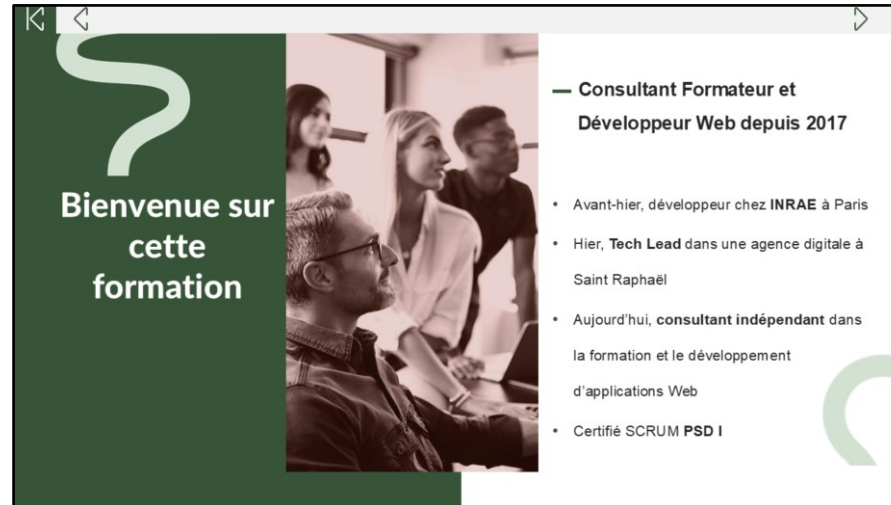


**SQL – Les fondamentaux**

**Du 06/10/2025 au 08/10/2025**

— Glodie Tshimini

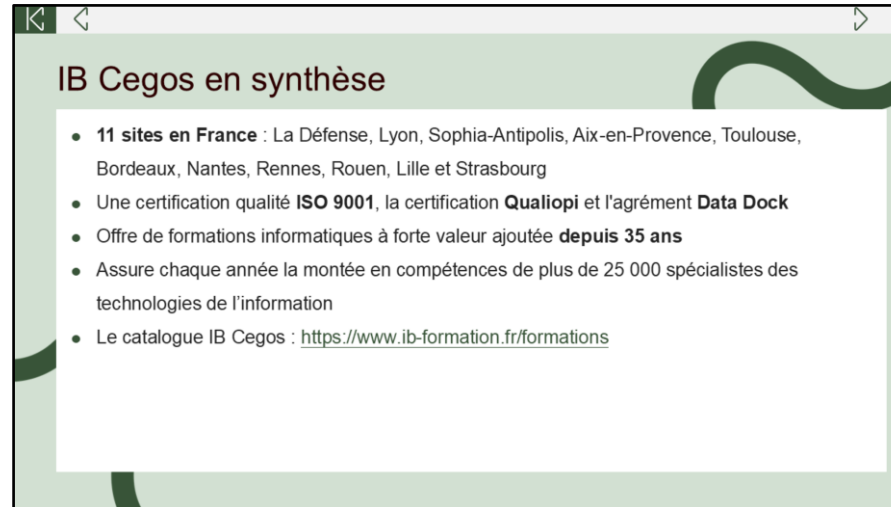
Ce support est la propriété d'IB SA



Bienvenue sur  
cette  
formation

— Consultant Formateur et  
Développeur Web depuis 2017


- Avant-hier, développeur chez **INRAE** à Paris
- Hier, **Tech Lead** dans une agence digitale à Saint Raphaël
- Aujourd'hui, **consultant indépendant** dans la formation et le développement d'applications Web
- Certifié **SCRUM PSD I**

A screenshot of a presentation slide with a light green background and dark green wavy borders. The title 'IB Cegos en synthèse' is in a dark red font. Below it is a bulleted list of five items. The slide is displayed in a window with navigation icons at the top.

### IB Cegos en synthèse

- **11 sites en France** : La Défense, Lyon, Sophia-Antipolis, Aix-en-Provence, Toulouse, Bordeaux, Nantes, Rennes, Rouen, Lille et Strasbourg
- Une certification qualité **ISO 9001**, la certification **Qualiopi** et l'agrément **Data Dock**
- Offre de formations informatiques à forte valeur ajoutée **depuis 35 ans**
- Assure chaque année la montée en compétences de plus de 25 000 spécialistes des technologies de l'information
- Le catalogue IB Cegos : <https://www.ib-formation.fr/formations>

Lien catalogue IB Cegos de formation informatique par thématique <https://www.ib-formation.fr/formations>



**Avant de commencer**

- Durée de la formation : 3 jours
- Objectifs :
  - ▶ Comprendre les principaux concepts des SGDBR (Système de Gestion des Bases de Données Relationnelles)
  - ▶ Appréhender l'écriture des requêtes SQL pour extraire et mettre à jour des données
  - ▶ Savoir extraire les informations de plusieurs tables
  - ▶ Assimiler les fonctions standards du langage SQL
- Organisation
  - ▶ Horaires : 9h00 à 17h00
  - ▶ Pauses : 15 min en matinée et après midi (vers 10h30 et 15h30)
  - ▶ Déjeuner : 1h00 (créneau 12h30– 13h30)



**Avant de commencer**

- A qui s'adresse cette formation
  - ▶ Développeurs
  - ▶ Architectes
  - ▶ Architectes
  - ▶ Administrateurs de bases de données
  - ▶ Exploitants intervenant sur un serveur de bases de données
- Prérequis
  - ▶ Maîtriser l'outil informatique et avoir des notions de gestion des données dans l'organisation
  - ▶ [Disposez-vous des compétences nécessaires pour suivre cette formation ? Testez-vous !](https://learninghub.ib-formation.com/prerequis/quiz-LA300-a.html)

Lien pour tester vos prérequis <https://learninghub.ib-formation.com/prerequis/quiz-LA300-a.html>

| Plan de la formation              |                                  |                                    |                                 |
|-----------------------------------|----------------------------------|------------------------------------|---------------------------------|
| Module 1 – Introduction           | Module 2 – Le LID                | Module 3 – Fonctions de calcul     | Module 5 – Sous-interrogations  |
| Rappels sur le modèle relationnel | La sélection des données         | Les fonctions de calcul            | Les sous-requêtes dans le where |
| Schéma de la BDD du stage         | Atelier 2.1                      | Atelier 3.1                        | Les sous-requêtes dans le from  |
| Caractéristiques du langage SQL   | Les conditions (ou restrictions) | Les clauses Group by et Having     | Atelier 5                       |
|                                   | Atelier 2.2                      | Atelier 3.2                        |                                 |
|                                   | Les tris                         |                                    |                                 |
|                                   | Atelier 2.3                      | Module 4 – Opérateurs ensemblistes | Module 6 – Le LMD               |
|                                   | Les jointures entre tables       | Les opérateurs ensemblistes        | Modifier les données            |
|                                   | Atelier 2.4                      | Atelier 4                          | Atelier 6                       |
|                                   |                                  |                                    | Module 7 – Notions sur le LDD   |

Plan de stage mis en ligne sur le site IB <https://www.ib-formation.fr/formations/bases-de-donnees/sql-les-fondamentaux> :

- 1. Introduction
  - Rappels sur le modèle relationnel
  - Les caractéristiques du langage SQL
- 2. Le Langage d'Interrogation de Données (LID)
  - La sélection de données
  - La gestion des valeurs null
  - Les restrictions ou conditions
  - Les tris
  - Les jointures
- 3- Utilisation des fonctions

- 4- Utilisation des opérateurs ensemblistes
  - Union
  - Intersect
  - Minus ou Except
- 5. Utilisation des sous interrogations
- 6. Le Langage de Manipulation de Données (LMD)
  - Le Insert
  - L'Update
  - Le Delete
- 7. Notions sur le langage de définition de données (LDD)
  - Création de tables : syntaxe
  - Types de données
  - Types de contraintes
  - Modifier la définition d'une table
  - Supprimer une table
  - Notions sur : Vue, index et synonyme



The image shows a slide titled "Plan de la formation prévisionnel" with a large green 'S' graphic on the left. The slide contains a table with two rows of three columns each, representing the schedule for three days (J 1, J 2, J 3) in the morning and afternoon.

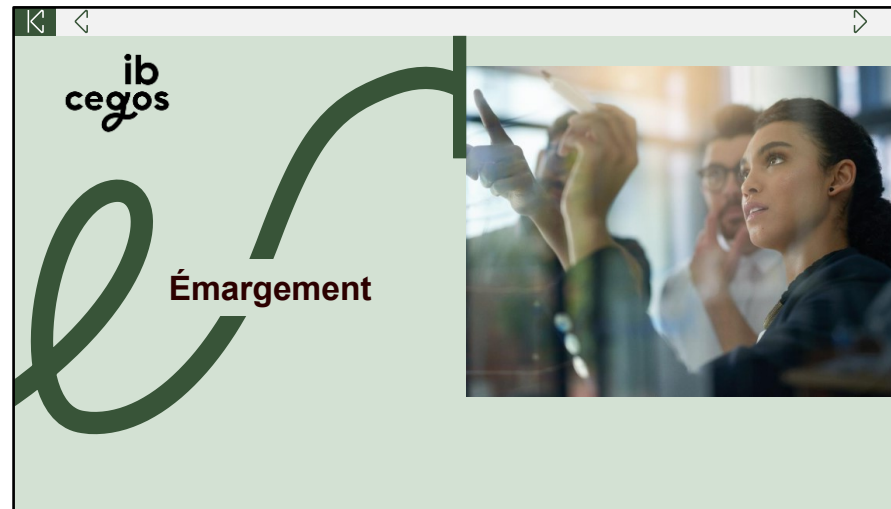
| J 1 - Matin                       | J 2 - Matin                         | J 3 - Matin                      |
|-----------------------------------|-------------------------------------|----------------------------------|
| Rappels sur le modèle relationnel | Daily & révisions de la journée 1   | Daily & Révisions jour 2         |
| Schéma de la BDD du stage         | Les jointures entre tables          | Les opérateurs ensemblistes      |
| Caractéristiques du langage SQL   | Atelier                             | Atelier                          |
|                                   |                                     | Utiliser des sous-interrogations |
| J 1 – Après midi                  | J 2 – Après midi                    | J 3 – Après midi                 |
| La sélection des données          | Les fonctions de calcul             | Atelier                          |
| Atelier                           | Atelier : fonctions de regroupement | Modifier les données             |
| Les conditions (ou restrictions)  | Les clauses Group by et Having      | Atelier                          |
| Atelier                           | Atelier                             | Notions sur le LDD               |
| Les tris                          |                                     |                                  |

Plan de stage mis en ligne sur le site IB <https://www.ib-formation.fr/formations/bases-de-donnees/sql-les-fondamentaux>:

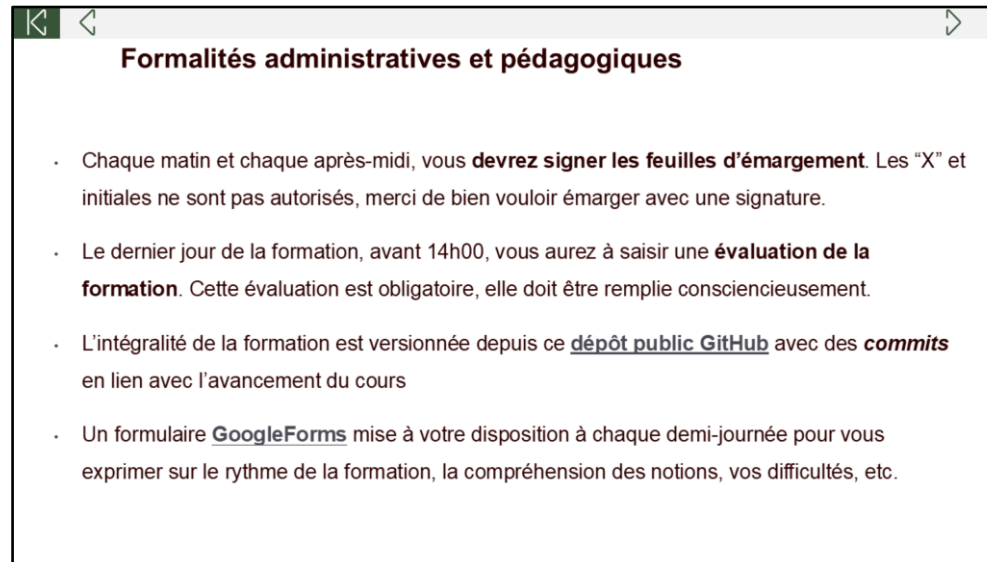
- 1. Introduction
  - Rappels sur le modèle relationnel
  - Les caractéristiques du langage SQL
- 2. Le Langage d'Interrogation de Données (LID)
  - La sélection de données
  - La gestion des valeurs null
  - Les restrictions ou conditions
  - Les tris
  - Les jointures
- 3. Utilisation des fonctions



- 4. Utilisation des opérateurs ensemblistes
  - Union
  - Intersect
  - Minus ou Except
- 5. Utilisation des sous interrogations
- 6. Le Langage de Manipulation de Données (LMD)
  - Le Insert
  - L'Update
  - Le Delete
- 7. Notions sur le langage de définition de données (LDD)
  - Création de tables : syntaxe
  - Types de données
  - Types de contraintes
  - Modifier la définition d'une table
  - Supprimer une table
  - Notions sur : Vue, index et synonyme



- Émargez **chaque début de demi-journée avec une vraie signature** en scannant le **QR Code** projeté dans la salle ou via le lien reçu par votre email d'inscription à cette formation.



The screenshot shows a presentation slide with a title bar at the top containing navigation icons (back, forward, search, etc.). The title of the slide is "Formalités administratives et pédagogiques". Below the title, there is a bulleted list of four items.

### Formalités administratives et pédagogiques

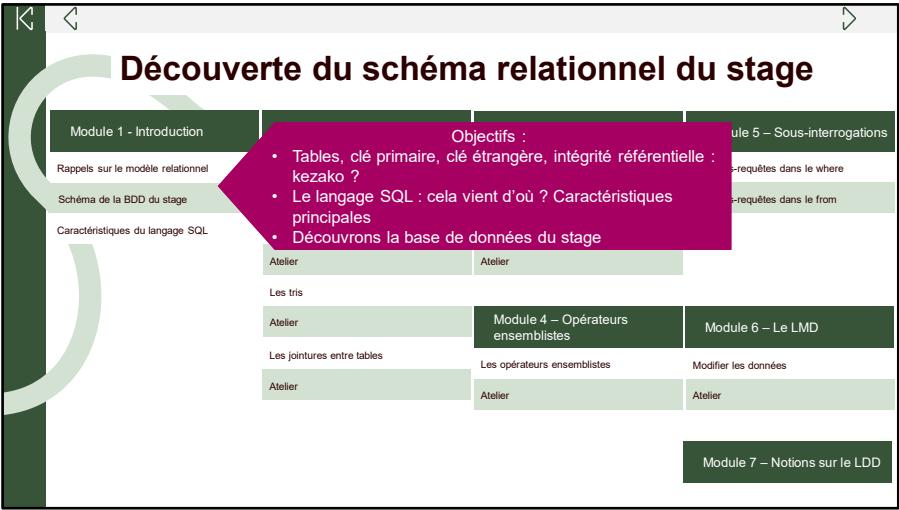
- Chaque matin et chaque après-midi, vous **devrez signer les feuilles d'émargement**. Les "X" et initiales ne sont pas autorisés, merci de bien vouloir émarger avec une signature.
- Le dernier jour de la formation, avant 14h00, vous aurez à saisir une **évaluation de la formation**. Cette évaluation est obligatoire, elle doit être remplie consciencieusement.
- L'intégralité de la formation est versionnée depuis ce [dépôt public GitHub](#) avec des **commits** en lien avec l'avancement du cours
- Un formulaire **GoogleForms** mise à votre disposition à chaque demi-journée pour vous exprimer sur le rythme de la formation, la compréhension des notions, vos difficultés, etc.

- Lien GitHub <https://github.com/glo10/06102025-sql>
- Lien Google Forms <https://docs.google.com/forms/d/e/1FAIpQLSeZI2mGPbWjofY4GNLZwf00xciAde28F2kywHWbICBoV6NVrQ/viewform?usp=header>

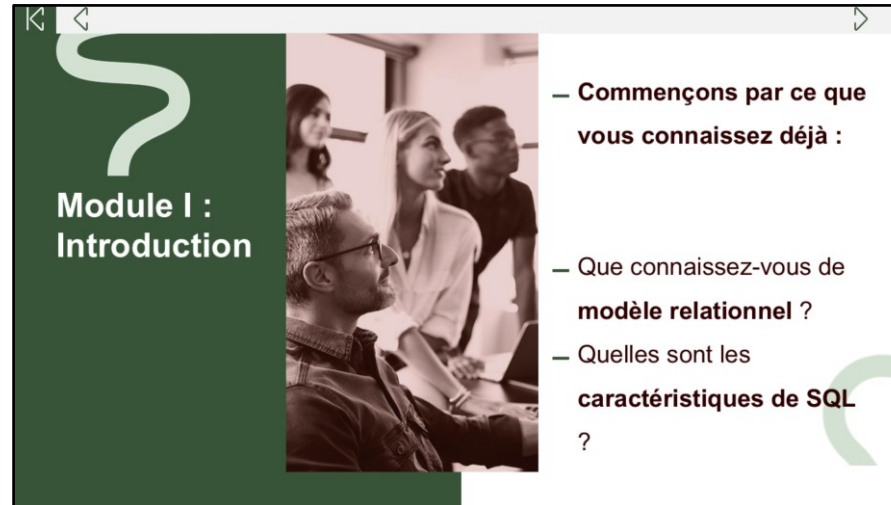


**Avant de commencer**

- Priorité aux échanges  
(durant les ateliers également)
- Tour de table :
  - ▶ Prénom
  - ▶ Nom
  - ▶ Société
  - ▶ Titre/fonction
  - ▶ Votre expérience avec SQL ?
  - ▶ Vos attentes vis-à-vis de cette formation ?




Plan prévisionnel de formation



The slide is titled "Module I : Introduction" and features a green background on the left with a white stylized 'S' logo. A photograph of three people (two women and one man) in a meeting is visible in the center. The right side of the slide contains a list of topics to be covered.

## Module I : Introduction

- Commençons par ce que vous connaissez déjà :
- Que connaissez-vous de **modèle relationnel** ?
- Quelles sont les **caractéristiques de SQL** ?



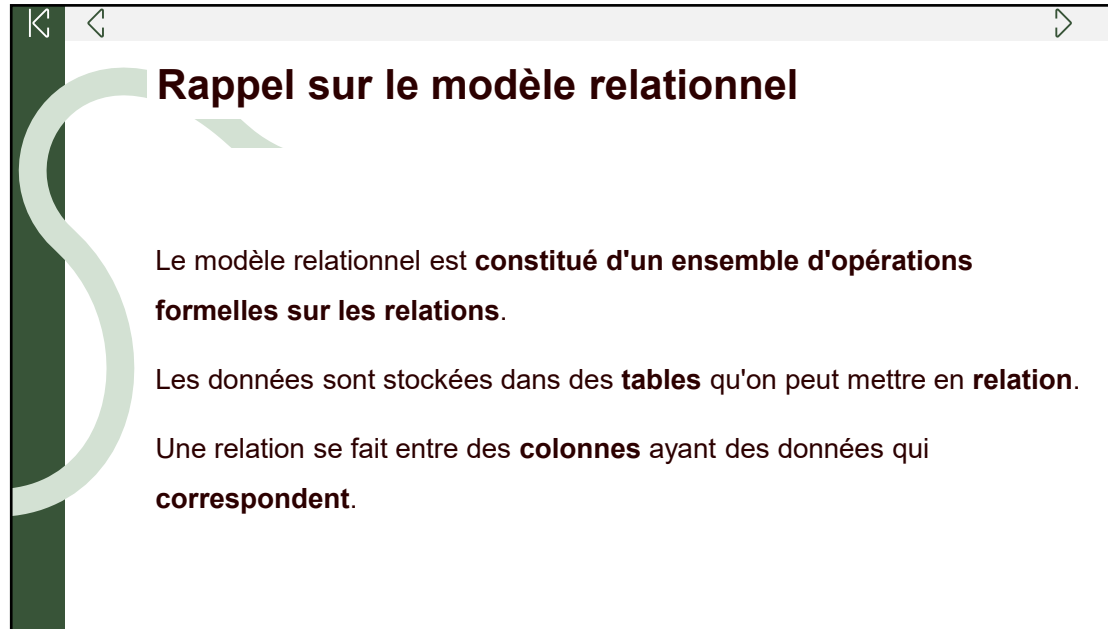
## Rappel sur le modèle relationnel

Une base de données est un **ensemble cohérent d'informations mémorisées sur support informatique**.

Ces informations sont **accessibles** à l'aide d'une application appelée **système de gestion de base de données (SGBD)**. Si ce SGBD est basé sur le modèle relationnel de CODD, on dit qu'il s'agit d'un système de gestion de base de données relationnel (SGBDR).

Pour **dialoguer** avec un **SGBDR** on utilise le **langage SQL**.  
Ce langage permet de soumettre des requêtes (des questions) au SGBDR.

Lien wikipédia Edgar Frank « Ted » Codd [https://fr.wikipedia.org/wiki/Edgar\\_Frank\\_Codd](https://fr.wikipedia.org/wiki/Edgar_Frank_Codd)



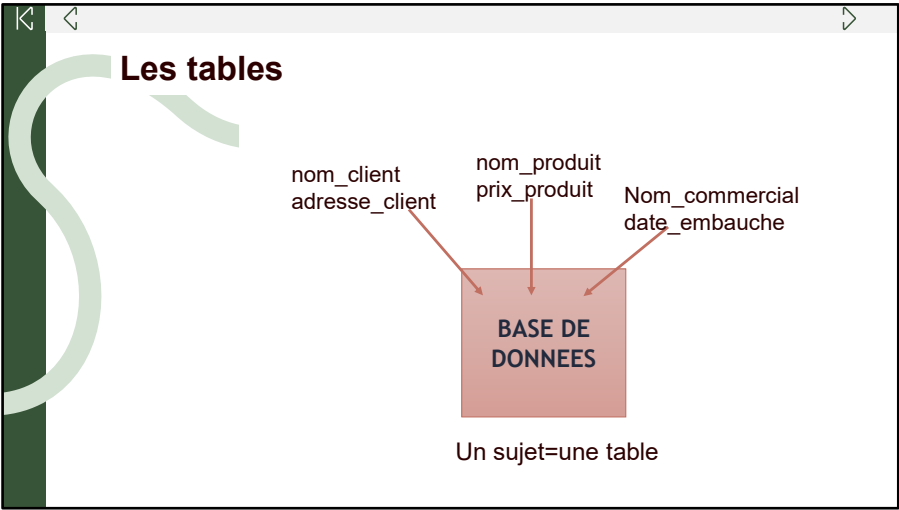
## Rappel sur le modèle relationnel

Le modèle relationnel est **constitué d'un ensemble d'opérations formelles sur les relations**.

Les données sont stockées dans des **tables** qu'on peut mettre en **relation**.

Une relation se fait entre des **colonnes** ayant des données qui **correspondent**.





# Rappel sur le modèle relationnel

La modélisation relationnelle permet de représenter les relations à l'aide de **tables** (à deux dimensions)

Une ligne d'une **table** représente donc une **entité**.

Un **attribut** est le **nom des colonnes** qui constitue la définition d'une table. Il comporte un **typage** de données.

On appelle **tuple** (ou n-uplet) une ligne de la table.

Attributs

| Marque  | Modèle | Série  | Numéro      |
|---------|--------|--------|-------------|
| Renault | 18     | RL     | 4698 SJ 45  |
| Peugeot | 309    | Chorus | 5647 ABY 82 |
| Ford    | Escort | Match  | 8562 EV 23  |

Tuples  
(N-uplets)

[Source image commentcamarche.net](#)

# Rappel sur le modèle relationnel

► La **cardinalité** d'une relation est le nombre de **tuples** qui la composent.

► La **clé principale (ou primaire)** d'une relation est l'**attribut**, ou l'**ensemble d'attributs**, permettant de désigner de façon **unique** un tuple.

► Une **clé étrangère**, par contre, est une clé faisant **référence** à une clé appartenant à une autre table.

▪ Le nom des colonnes à relier peut être différent, mais le **type de données doit être le même**.

Produits

|           |              |
|-----------|--------------|
| Référence | chaîne       |
| Nom       | chaîne       |
| En stock  | booléen      |
| Prix (€)  | réel positif |

Ventes

|                   |                |
|-------------------|----------------|
| Numéro            | chaîne         |
| Référence produit | chaîne         |
| Date              | chaîne         |
| Quantité          | entier positif |

clé primaire

clé étrangère

lien

domaines

attributs

Source image [info.blaisepacal.fr](http://info.blaisepacal.fr)

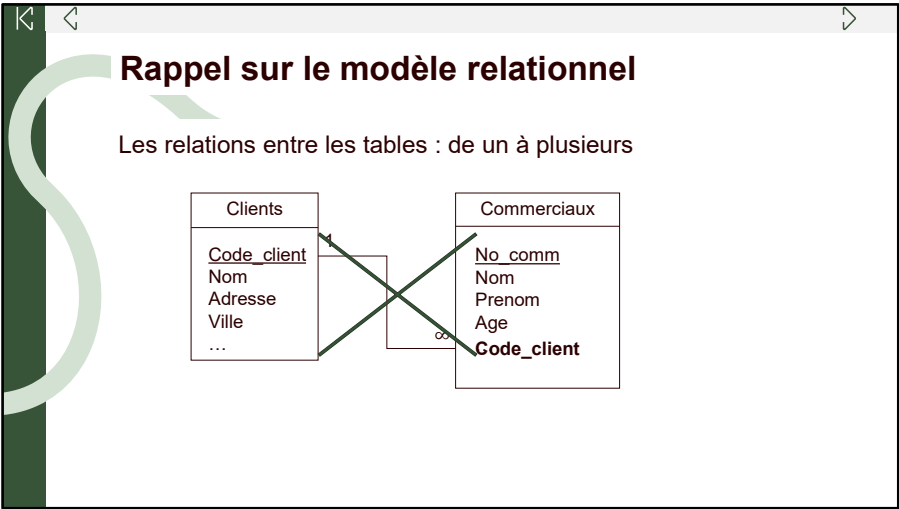
Formation - SQL les fondamentaux avec Glodie Tshimini

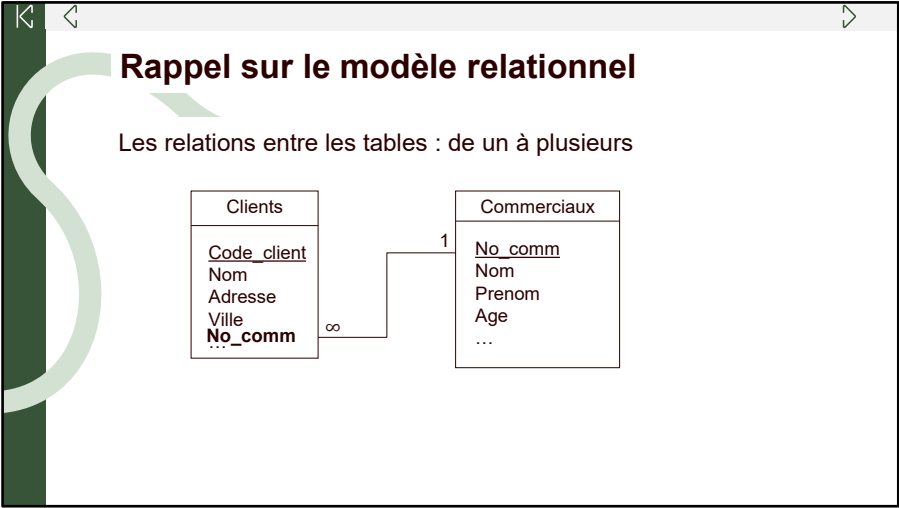
20

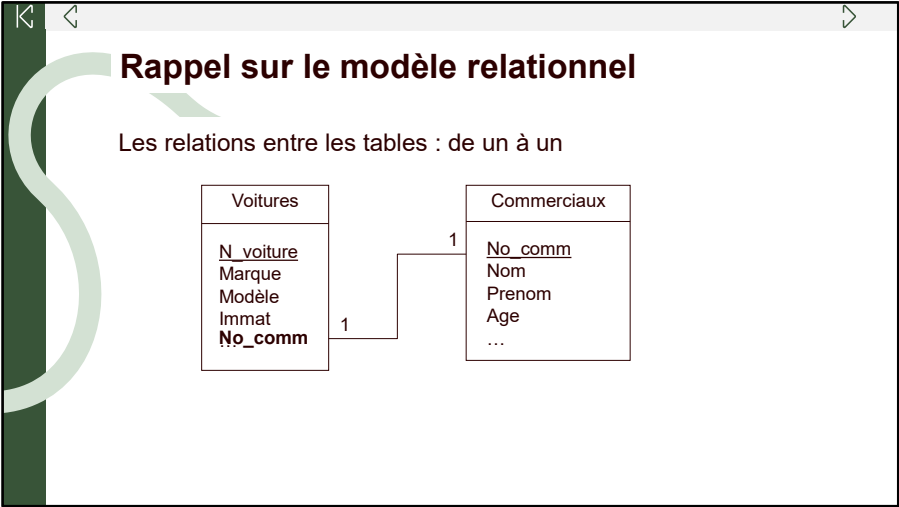
## Rappel sur le modèle relationnel

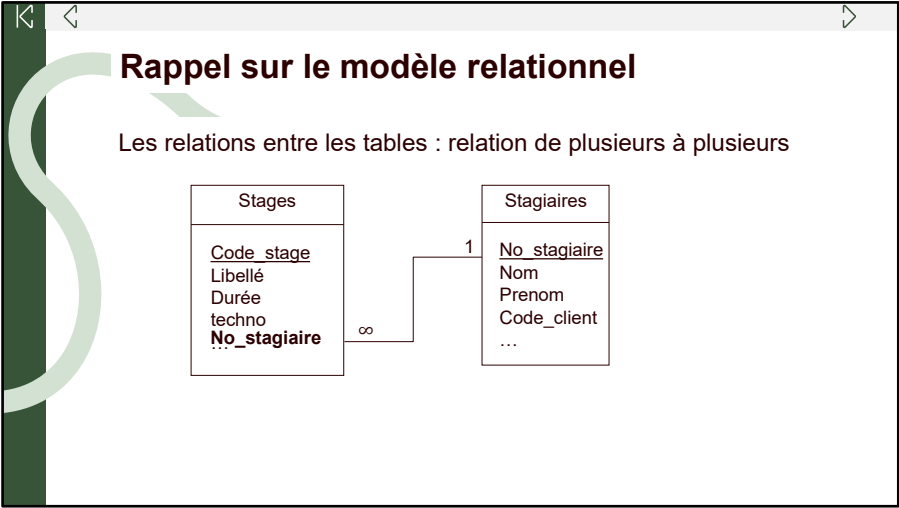
Les données sont stockées dans des tables

|                   |   |  |
|-------------------|---|--|
| Employes          | ← | Nom de la table                                |
| <u>No_employe</u> | ← | Clé primaire (identifiant unique d'une table ) |
| Nom               | ← | Colonnes (ou champs ou attributs)              |
| Prenom            | ← |  |
| Fonction          |   |  |

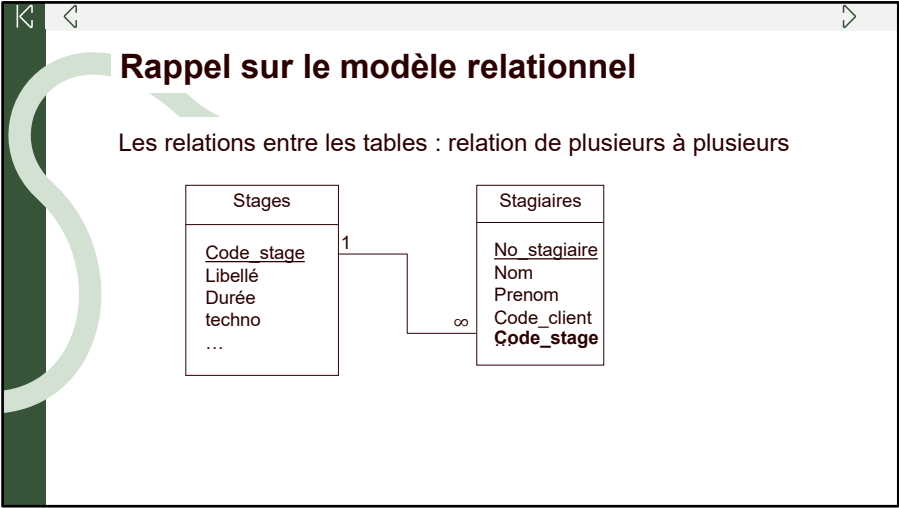


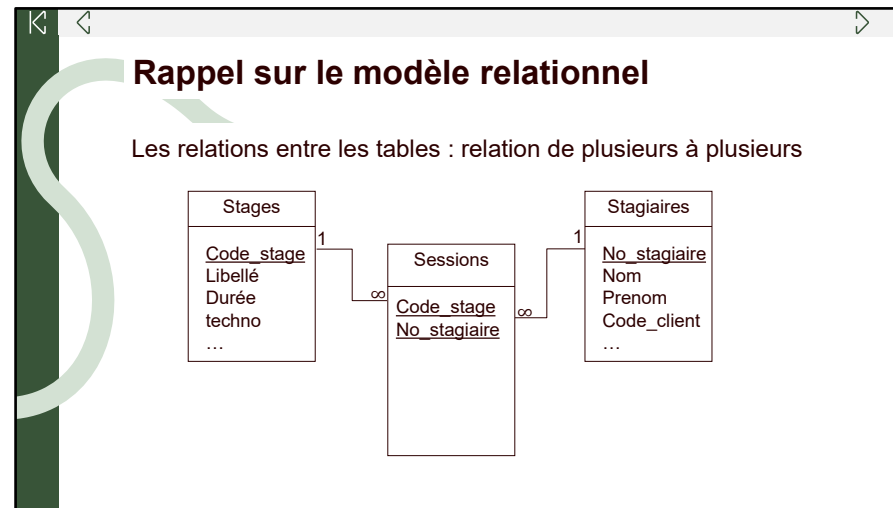






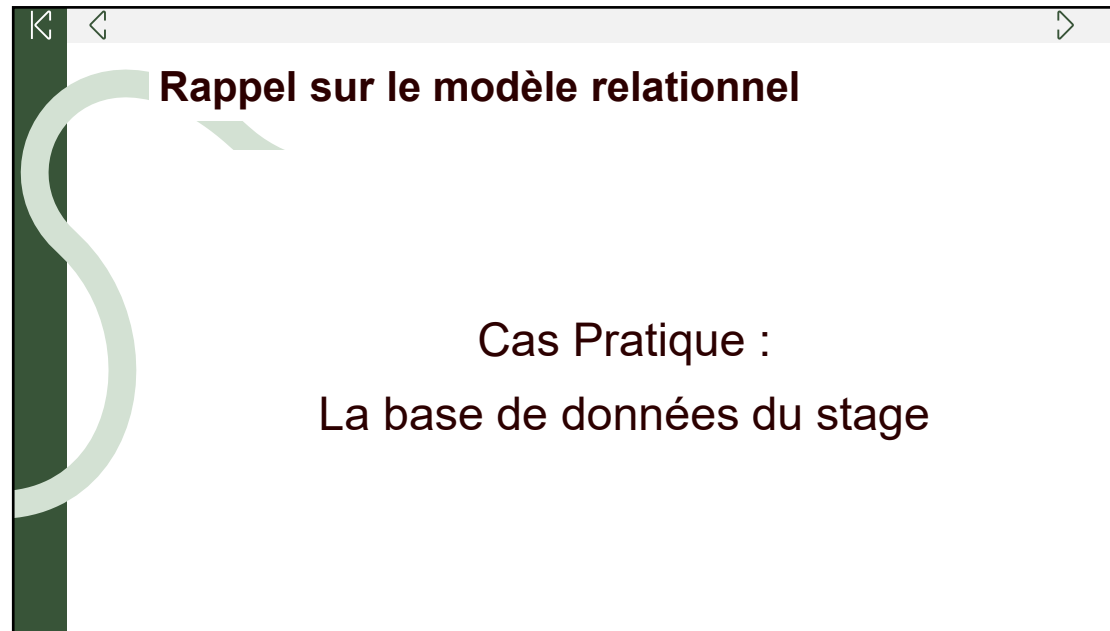




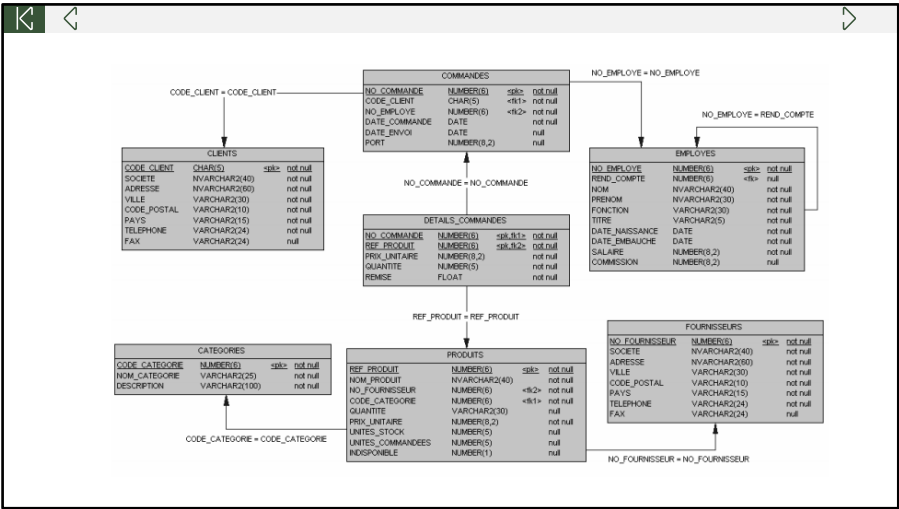


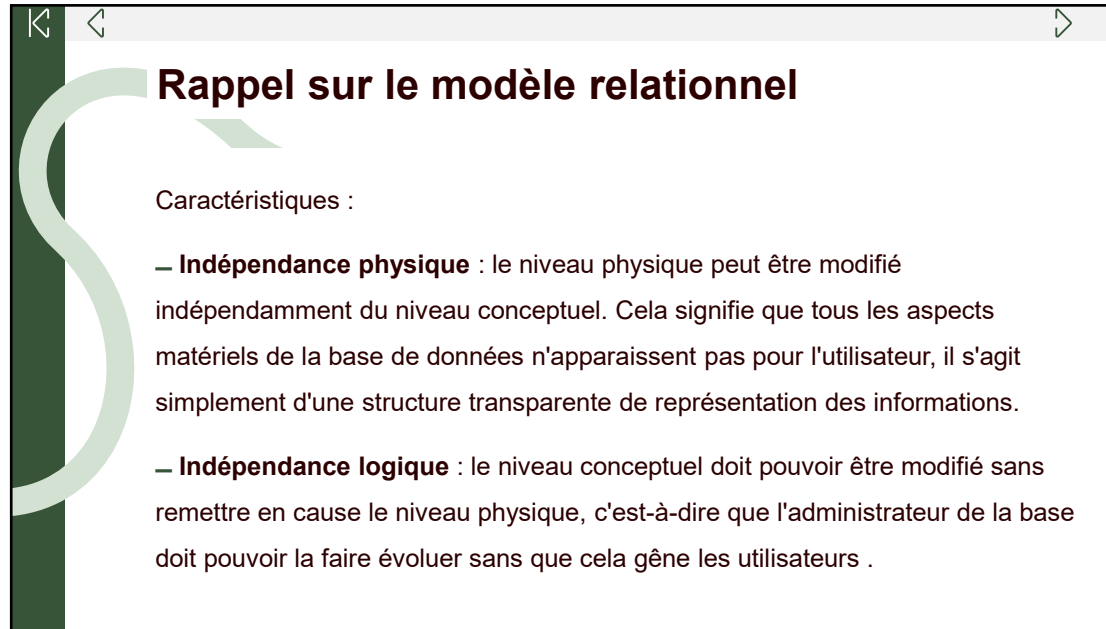
- **L'intégrité référentielle permet de garantir les relations entre les tables et la cohérence des données.**
- **Elle contrôle la cohérence des données à l'insertion, modification et suppression des données**
- Concrètement par les contraintes d'unicité par l'utilisation des clés primaires et clés étrangères
  - Rappel, une clé primaire (PK) identifie une ligne de façon unique dans une table
  - Une clé étrangère fait référence (FK) la clé primaire ou une clé unique d'une autre table
- Donc lorsqu'une clé étrangère existe sur une table, l'intégrité référentielle impose d'avoir une valeur existante dans l'autre table référencée.
- Par exemple, on ne peut pas insérer une commande avec un identifiant client dans la table commande qui fait référence à un identifiant client si ce dernier n'existe pas.
- Même raisonnement pour la modification d'une valeur qui stocke une clé étrangère.

- Pour la suppression d'une ligne qui contient une clé étrangère, par défaut, l'opération est refusée pour éviter d'avoir des enregistrements orphelins qui ne font pas référence à un PK, cependant, il est possible de changer ce comportement avec les contraintes ON DELETE CASCADE, ON DELETE SET NULL.



Nous allons utiliser un extrait d'une base de données classique **Northwind** représentant une entreprise commerciale de produits alimentaires avec des clients, fournisseurs, employés et des commandes.






## Rappel sur le modèle relationnel

Caractéristiques :

- **Indépendance physique** : le niveau physique peut être modifié indépendamment du niveau conceptuel. Cela signifie que tous les aspects matériels de la base de données n'apparaissent pas pour l'utilisateur, il s'agit simplement d'une structure transparente de représentation des informations.
- **Indépendance logique** : le niveau conceptuel doit pouvoir être modifié sans remettre en cause le niveau physique, c'est-à-dire que l'administrateur de la base doit pouvoir la faire évoluer sans que cela gêne les utilisateurs .

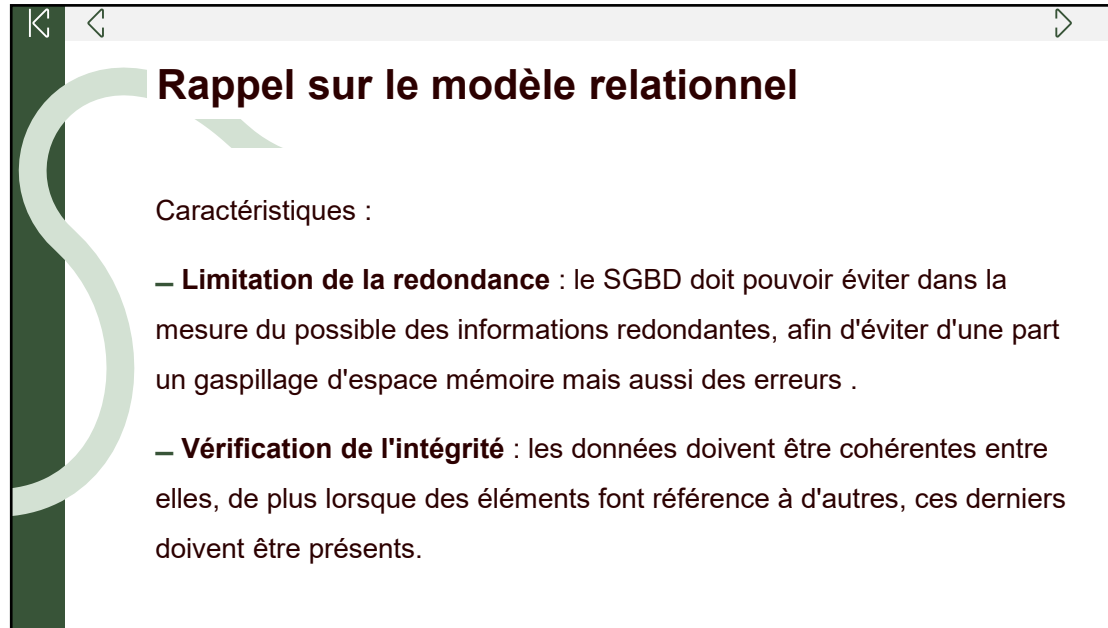
- Séparation et indépendance entre la logique conceptuelle des données et sa mise en place physique avec une infrastructure matériels



## Rappel sur le modèle relationnel

Caractéristiques :

- **Manipulabilité** : des personnes ne connaissant pas la base de données doivent être capables de décrire leur requête sans faire référence à des éléments techniques de la base de données.
- **Rapidité des accès** : le système doit pouvoir fournir les réponses aux requêtes le plus rapidement possible, cela implique des algorithmes de recherche rapides.
- **Administration centralisée** : le SGBD doit permettre à l'administrateur de pouvoir manipuler les données, insérer des éléments, vérifier son intégrité de façon centralisée.




## Rappel sur le modèle relationnel

Caractéristiques :

- **Limitation de la redondance** : le SGBD doit pouvoir éviter dans la mesure du possible des informations redondantes, afin d'éviter d'une part un gaspillage d'espace mémoire mais aussi des erreurs .
- **Vérification de l'intégrité** : les données doivent être cohérentes entre elles, de plus lorsque des éléments font référence à d'autres, ces derniers doivent être présents.





## Rappel sur le modèle relationnel

Caractéristiques :

- **Partageabilité des données** : le SGBD doit permettre l'accès simultané à la base de données par plusieurs utilisateurs.
- **Sécurité des données** : le SGBD doit présenter des mécanismes permettant de gérer les droits d'accès aux données selon les utilisateurs.




## Rappel sur le modèle relationnel

Une **requête** est un **ordre adressé à un SGBD**. Cet ordre peut consister à **extraire**, à **ajouter**, à **modifier**, à **administrer** les données de la base.

De façon générale, l'utilisateur, comme l'administrateur, dialogue avec le SGBD en lui soumettant des requêtes (des questions) et en récupérant en retour des résultats (les réponses).

- **Une requête = Une question**
- **Une requête attend en retour une réponse de la base de données**





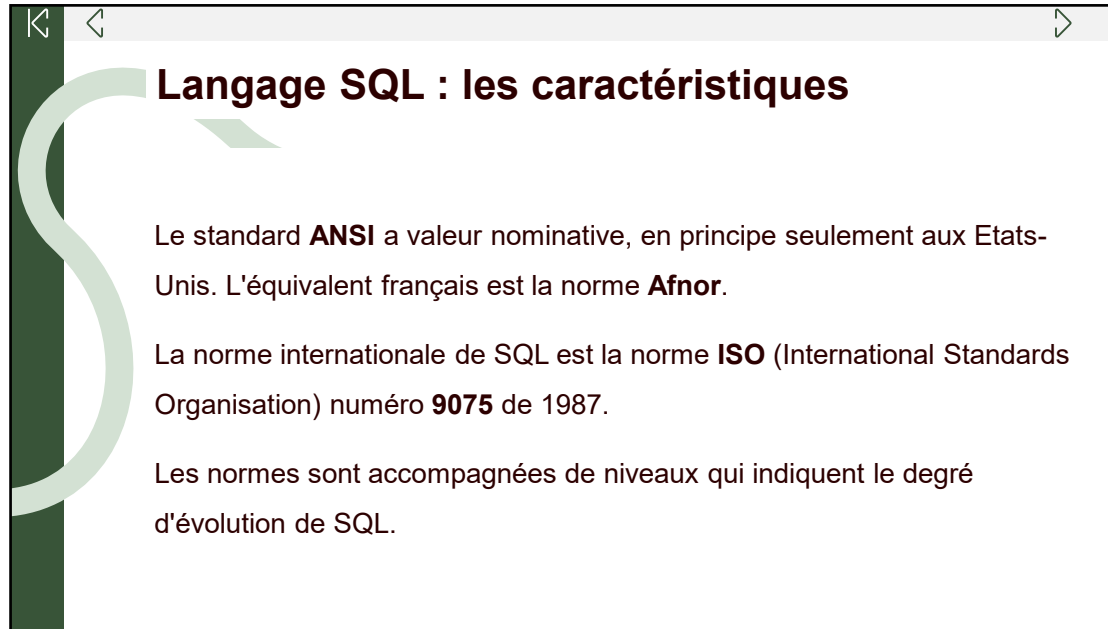
The screenshot shows a presentation slide with a dark green vertical bar on the left containing a large, light green stylized letter 'S'. The slide title is 'Langage SQL : les caractéristiques'. The text on the slide describes SQL as a standard in relational interfaces, developed by IBM at San José, based on explicit English keywords, and normalized. It also mentions its origin from SEQUEL.

## Langage SQL : les caractéristiques

Le langage SQL est devenu le **standard en matière d'interface relationnelle**, ceci probablement parce que ce langage est :

- issu de SEQUEL (interface de System-R), SQL a été **développé chez IBM** à San José ... !
- basé sur des mots clefs anglais explicites, il est relativement simple et facile à apprendre pour des utilisateurs non-informaticiens. Il illustre bien la tendance des langages formels à s'orienter vers un certain "**langage naturel**".
- **normalisé**

- SEQUEL = 1970
- Langage SQL = 1975
- Consultez ce lien pour avoir les détails sur l'histoire de SQL <https://learnsql.fr/blog/l-histoire-de-sql-comment-tout-a-commence/>

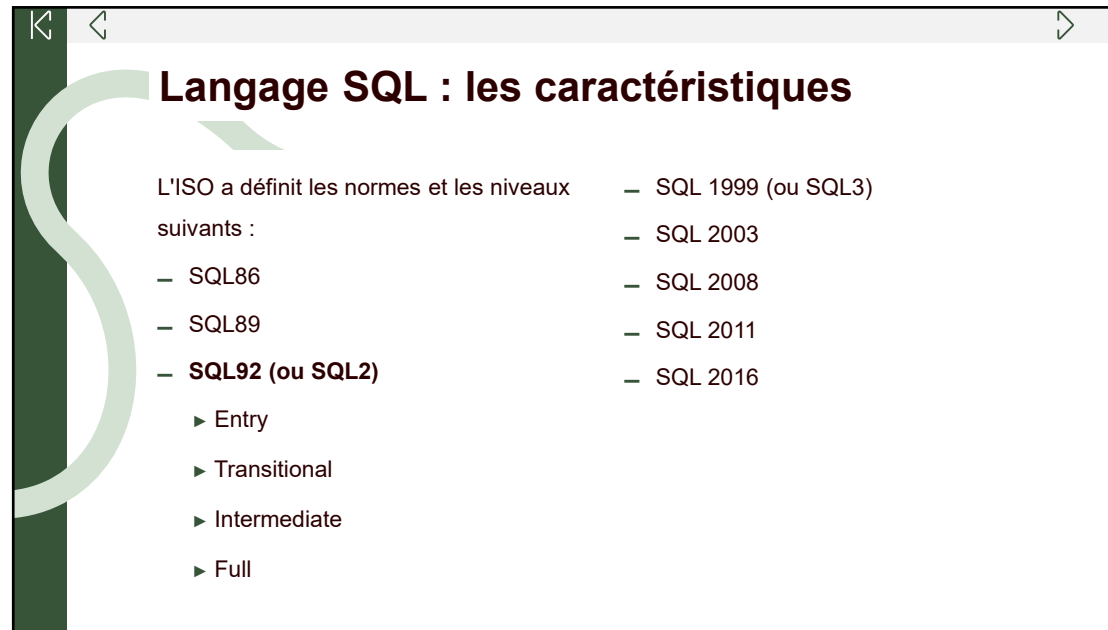


**Langage SQL : les caractéristiques**

Le standard **ANSI** a valeur nominative, en principe seulement aux Etats-Unis. L'équivalent français est la norme **Afnor**.

La norme internationale de SQL est la norme **ISO** (International Standards Organisation) numéro **9075** de 1987.

Les normes sont accompagnées de niveaux qui indiquent le degré d'évolution de SQL.



## Langage SQL : les caractéristiques

L'ISO a défini les normes et les niveaux suivants :

- SQL86
- SQL89
- **SQL92 (ou SQL2)**
  - ▶ Entry
  - ▶ Transitional
  - ▶ Intermediate
  - ▶ Full
- SQL 1999 (ou SQL3)
- SQL 2003
- SQL 2008
- SQL 2011
- SQL 2016

ANSI : American National Standard Institute

SQL est normalisé depuis **1986**, consultez ce lien pour avoir plus de détails et l'organigramme

<https://learnsql.fr/blog/l-histoire-des-normes-sql/>

- SQL86 = première norme, ratification par l'Iso
- SQL89 = révisions mineures
- SQL92 (ou SQL2) = révisions majeures, Norme la plus importante : *Join* ; les opérateurs ensemblistes ; *case...when* ; *alter* et *drop table* et *view*
- SQL 1999 (ou SQL3) = récursivité, déclencheurs, langage procédural, *rollup*
- SQL 2003 = prise en compte du xml, fonctions de fenêtrage, séquences, colonnes d'identité
- SQL 2006 = révisions mineures : intégration plus poussée avec XML (Xquery)
- SQL 2008 = révisions mineures. limitation du nombre de lignes (Offset/Fecht)
- SQL 2011, 2016 : révisions mineures



## Langage SQL : les caractéristiques

La norme définit **deux langages SQL** :

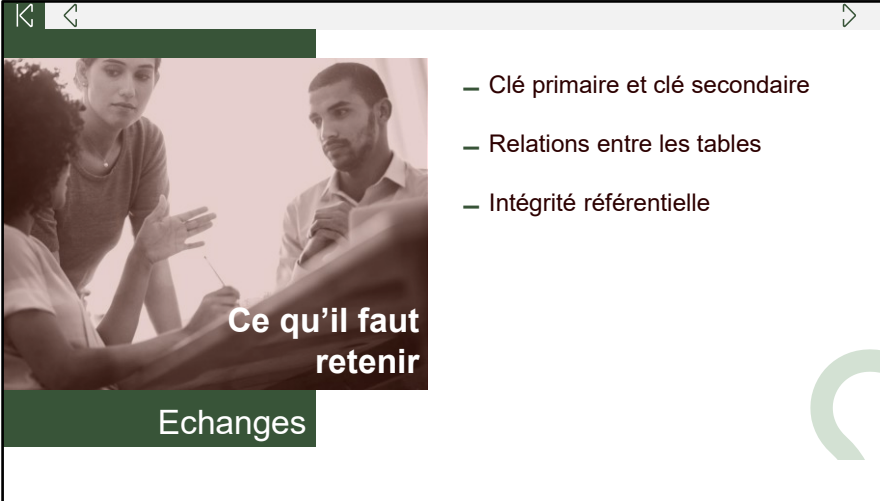
- un **Langage de Manipulation de Données et de modules**, (en anglais **SQLDML**), pour **déclarer** les **procédures** d'exploitation et les appels à utiliser dans les programmes.  
On peut également rajouter une composante pour l'interrogation de la base : le Langage d'Interrogation de Données.
- un **Langage de Définition de Données** (en anglais **SQL-DDL**), à utiliser pour **déclarer** les **structures** logiques de données et leurs **contraintes** d'intégrité. On peut également rajouter une composante pour la **gestion des accès aux données** :  
le **Langage de Contrôle de Données** (en anglais **SQL-DCL**)

# Langage SQL : les caractéristiques

Les instructions incontournables de SQL sont les suivantes :

| Langage SQL |        |        |          |
|-------------|--------|--------|----------|
| LMD         |        | LDD    |          |
| LID         |        | LCD    |          |
| SELECT      | INSERT | GRANT  | CREATE   |
|             | UPDATE | REVOKE | ALTER    |
|             | DELETE |        | TRUNCATE |
|             |        |        | DROP     |
|             |        |        | RENAME   |

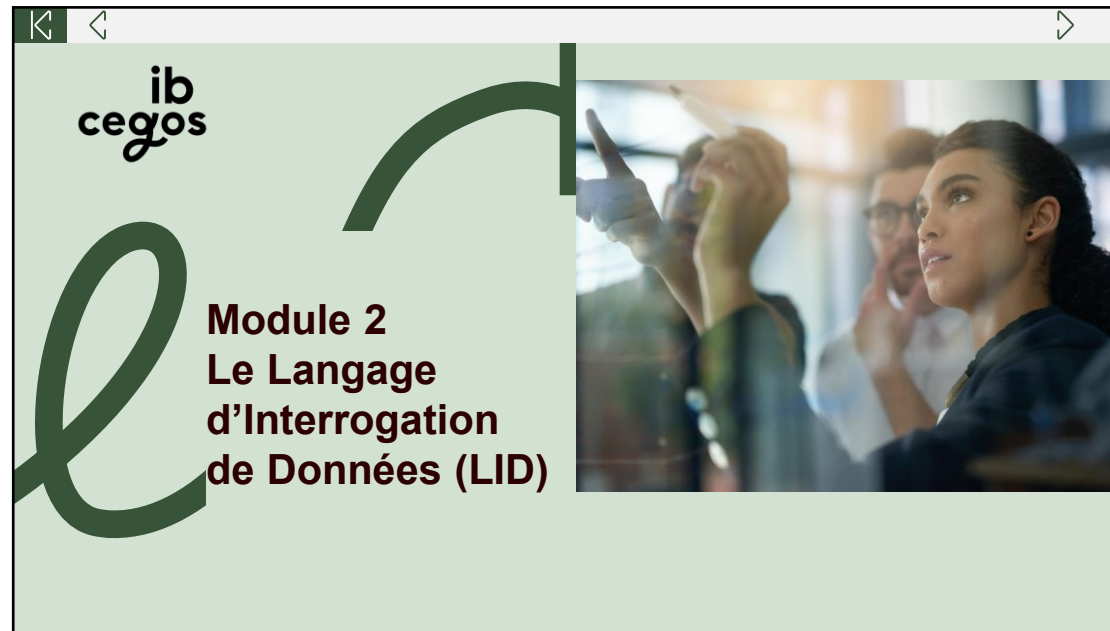




**Ce qu'il faut retenir**

**Echanges**

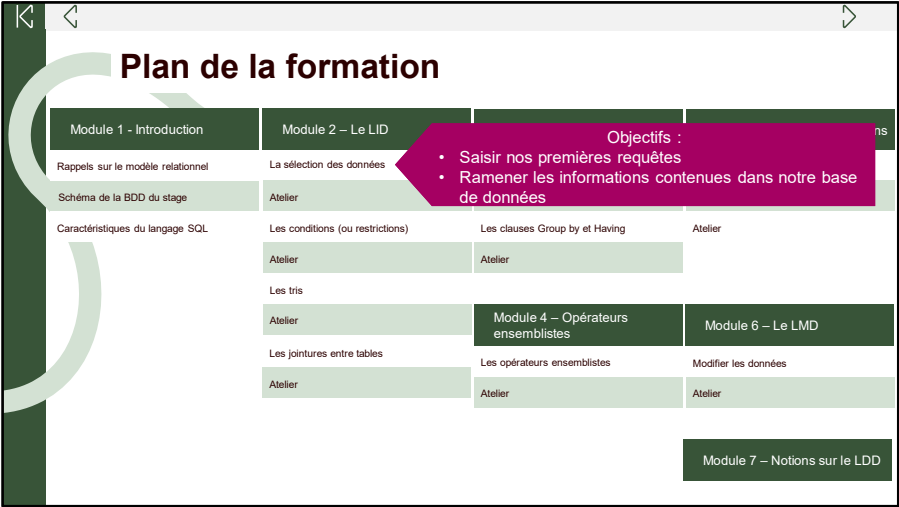
- Clé primaire et clé secondaire
- Relations entre les tables
- Intégrité référentielle

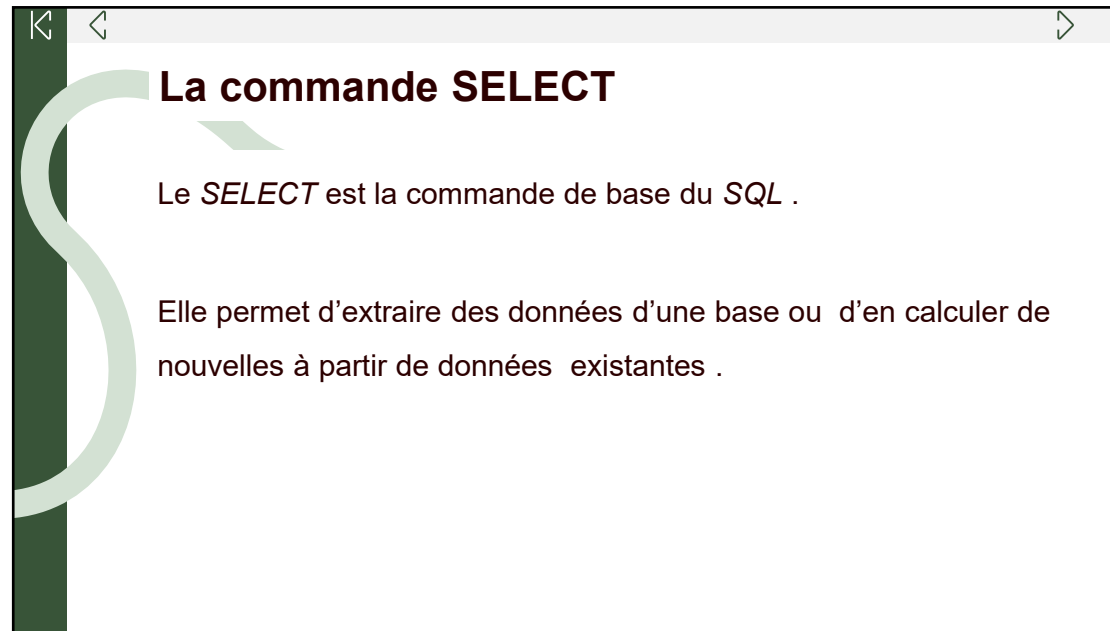


**Le Langage d'Interrogation des Données (LID) : les bases**

- La Sélection de données
- Gestion des valeurs *null*
- Les restrictions ou conditions
- Les tris
- Les jointures

A large green bracket is positioned to the left of the last three items in the list, grouping them together.



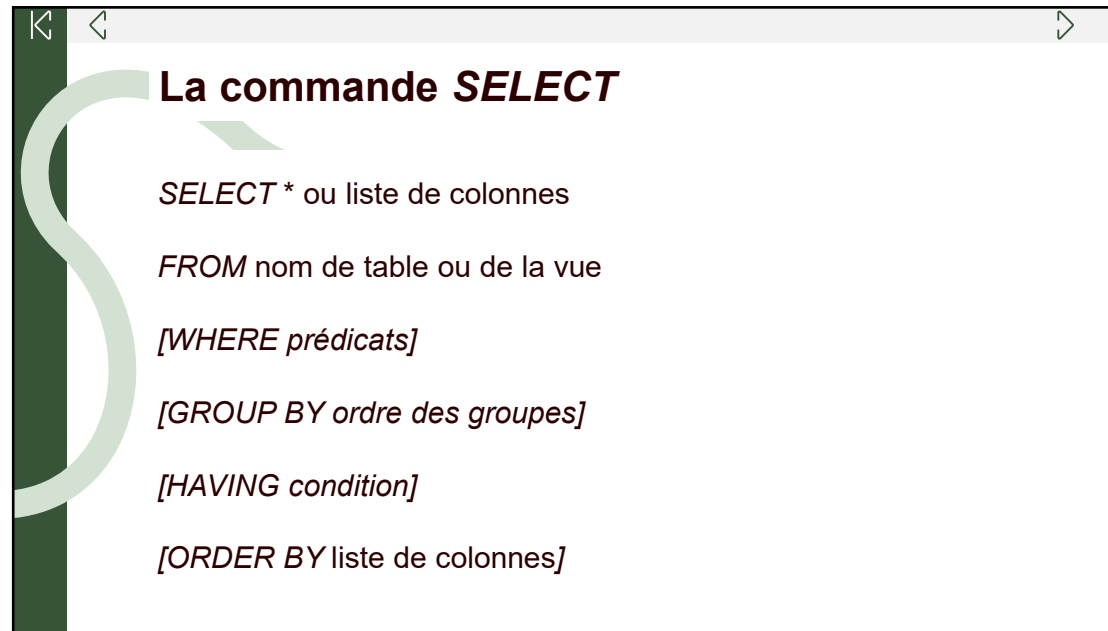


## La commande **SELECT**

Le *SELECT* est la commande de base du *SQL* .

Elle permet d'extraire des données d'une base ou d'en calculer de nouvelles à partir de données existantes .

Démonstration avec les différents SGBDR

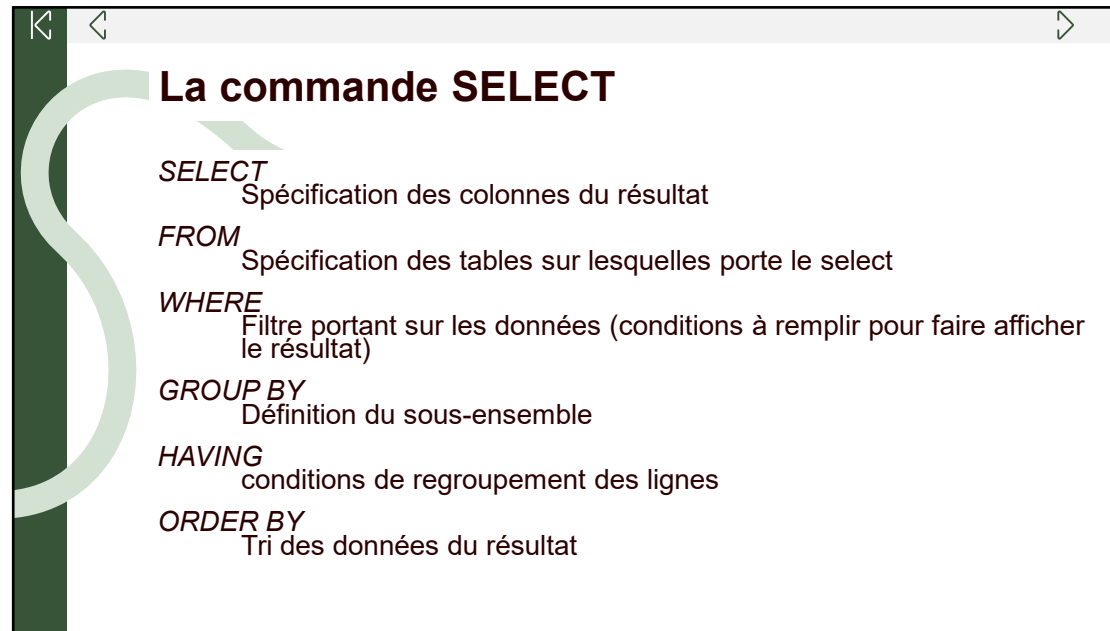


## La commande **SELECT**

- SELECT* \* ou liste de colonnes
- FROM* nom de table ou de la vue
- [WHERE prédicats]*
- [GROUP BY ordre des groupes]*
- [HAVING condition]*
- [ORDER BY liste de colonnes]*

[...] : les crochets dans ce support signifient que ce qui est entre crochet est facultatif, En aucun cas, il ne faut saisir ces crochets dans ses requêtes.

Les utilisateurs de Microsoft SQL Server peuvent voir des crochets dans les requêtes, mais ceux-ci encadrent le nom des colonnes et le nom des tables



## La commande **SELECT**

- SELECT**  
Spécification des colonnes du résultat
- FROM**  
Spécification des tables sur lesquelles porte le select
- WHERE**  
Filtre portant sur les données (conditions à remplir pour faire afficher le résultat)
- GROUP BY**  
Définition du sous-ensemble
- HAVING**  
conditions de regroupement des lignes
- ORDER BY**  
Tri des données du résultat

## La commande SELECT

Requête:

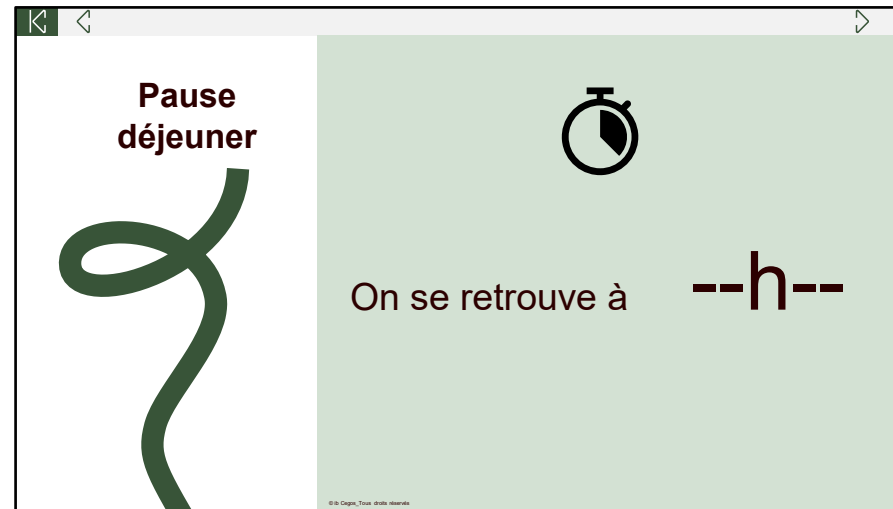
```
SELECT CLI_NOM, CLI_PRENOM FROM T_CLIENT;
```

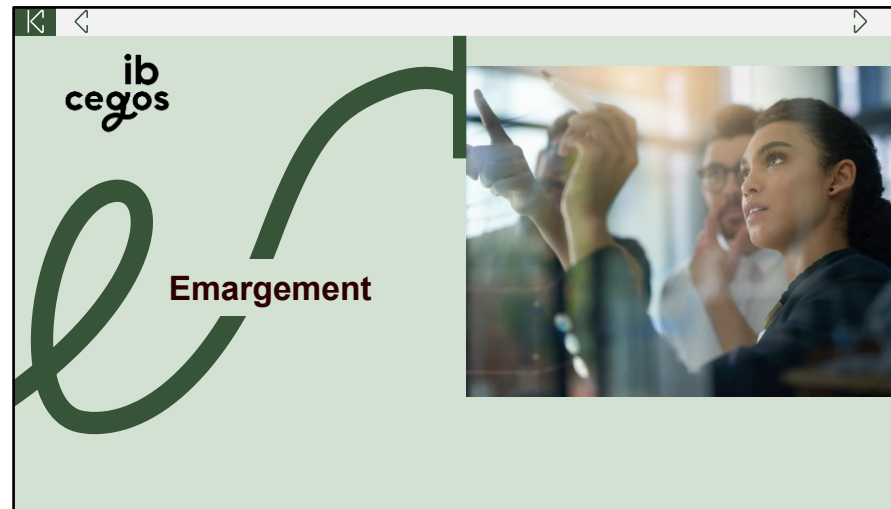
Résultat :

| CLI_NOM  | CLI_PRENOM    |
|----------|---------------|
| DUPONT   | Alain         |
| MARTIN   | Marc          |
| BOUVIER  | Alain         |
| DUBOIS   | Paul          |
| DREYFUS  | Jean          |
| FAURE    | Alain         |
| PAUL     | Marcel        |
| DUVAL    | Arsène        |
| PHILIPPE | André         |
| CHABAUD  | Daniel        |
| BAILLY   | Jean-François |
| ...      |               |

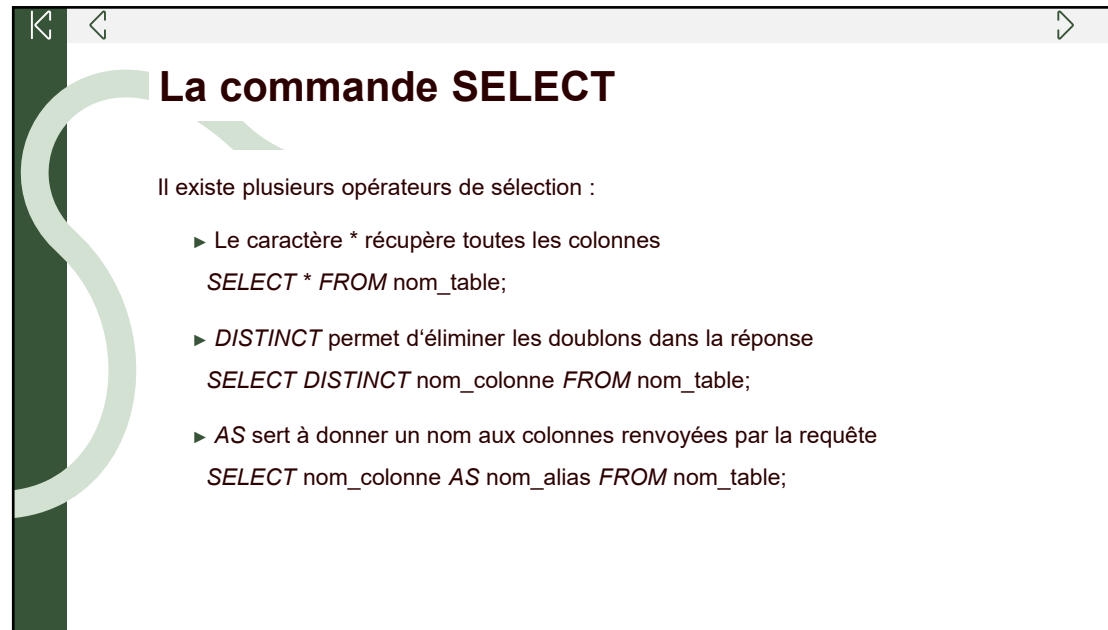
Le point-virgule permet de marquer explicitement la fin de la requête







Émargez **chaque début de demi-journée avec une vraie signature** en scannant le **QR Code** projeté dans la salle ou via le lien reçu par votre email d'inscription à cette formation.



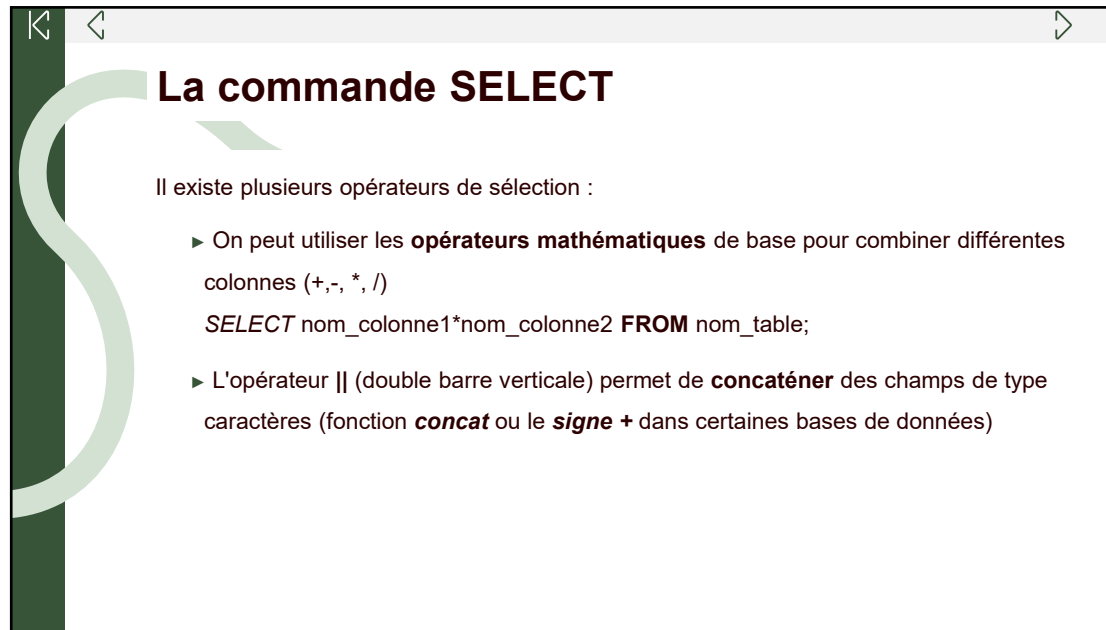
The screenshot shows a presentation slide with a dark green vertical bar on the left containing a large white letter 'S'. The slide title is 'La commande SELECT'. Below the title, it states 'Il existe plusieurs opérateurs de sélection :'. Three bullet points follow, each with a SQL example: 1. 'Le caractère \* récupère toutes les colonnes' with example 'SELECT \* FROM nom\_table;'. 2. 'DISTINCT permet d'éliminer les doublons dans la réponse' with example 'SELECT DISTINCT nom\_colonne FROM nom\_table;'. 3. 'AS sert à donner un nom aux colonnes renvoyées par la requête' with example 'SELECT nom\_colonne AS nom\_alias FROM nom\_table;'.

## La commande SELECT

Il existe plusieurs opérateurs de sélection :

- ▶ Le caractère \* récupère toutes les colonnes  
`SELECT * FROM nom_table;`
- ▶ **DISTINCT** permet d'éliminer les doublons dans la réponse  
`SELECT DISTINCT nom_colonne FROM nom_table;`
- ▶ **AS** sert à donner un nom aux colonnes renvoyées par la requête  
`SELECT nom_colonne AS nom_alias FROM nom_table;`

1. `SELECT * FROM categories;` : liste toutes les colonnes et les lignes de la table catégorie
2. `SELECT code_categorie, nom_categorie FROM categories;` : liste toutes les lignes de la table *categories* avec uniquement les colonnes *code\_categorie* et *nom\_categorie*
3. `SELECT code_categorie, nom_categorie AS "nom de la catégorie" FROM categories;`
  - Ici la colonne *nom\_categorie* est renommé à l'affichage par "nom de la catégorie"
4. `SELECT no_employe FROM commandes;`
5. `SELECT DISTINCT no_employe FROM employes;` : **DISTINCT** **supprime** les lignes identiques (**doublons**),
6. `SELECT "Bonjour", prenom FROM employes;`
  - "Bonjour" est une constante littérale, autrement dit une colonne fixe (n'est pas une colonne de la table)
  - Elle sera identique pour toutes les lignes retournées par la base de données
  - Sert à enrichir les résultats sans modifier la structure de la table, tester, ajouter une colonne supplémentaire dans l'affichage, etc.



## La commande SELECT

Il existe plusieurs opérateurs de sélection :

- ▶ On peut utiliser les **opérateurs mathématiques** de base pour combiner différentes colonnes (+, -, \*, /)  
`SELECT nom_colonne1*nom_colonne2 FROM nom_table;`
- ▶ L'opérateur || (double barre verticale) permet de **concaténer** des champs de type caractères (fonction **concat** ou le **signe +** dans certaines bases de données)

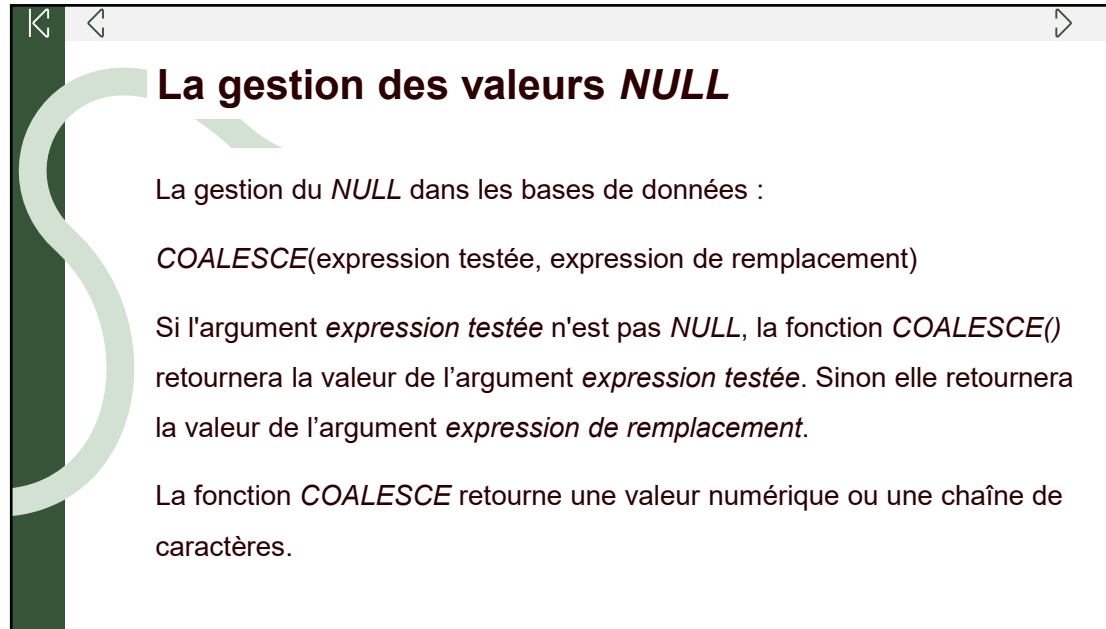
Des exemples opérations mathématiques et concaténation (mettre des informations bout à bout) :

1. `SELECT salaire * 1.10 FROM employes ;`
2. Concaténation Avec ORACLE et PostgreSQL
  1. `SELECT nom || " " || prenom FROM employes;`
3. Concaténation avec SQL Server
  1. `SELECT nom + " " + prenom FROM employes;`
4. Concaténation avec MYSQL et SQL SERVER
  1. `SELECT CONCAT(nom, " ", prenom) FROM employes;`

**Le Langage d'Interrogation des Données (LID) : les bases**

- La Sélection de données
- Gestion des valeurs *null*
- Les restrictions ou conditions
- Les tris
- Les jointures

A large green bracket is positioned to the left of the last three items in the list, grouping them together.



## La gestion des valeurs *NULL*

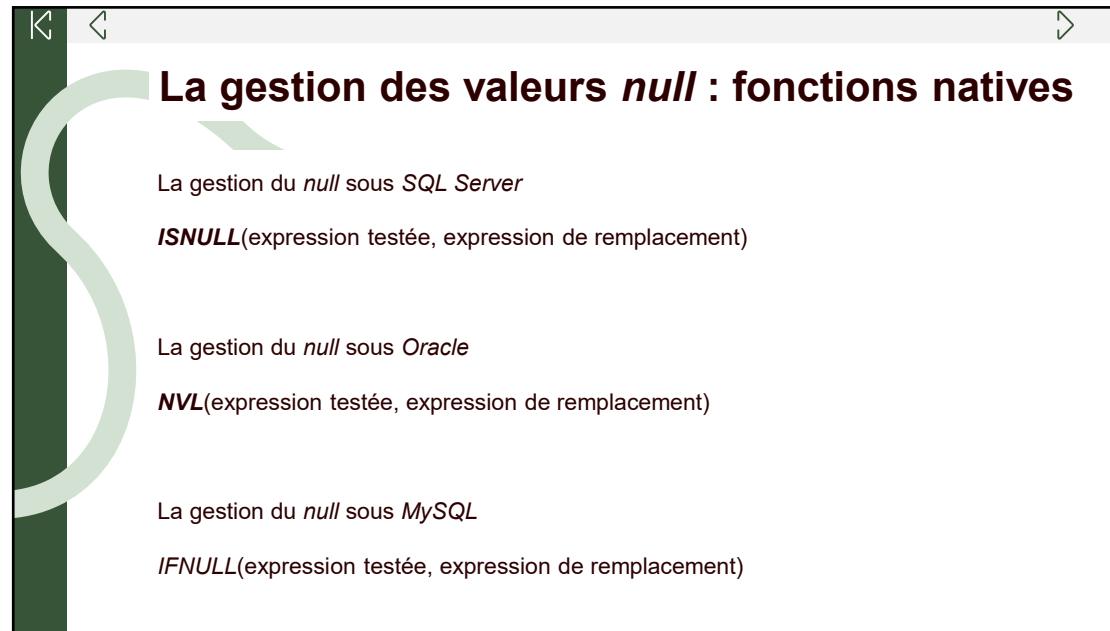
La gestion du *NULL* dans les bases de données :

*COALESCE*(expression testée, expression de remplacement)

Si l'argument *expression testée* n'est pas *NULL*, la fonction *COALESCE*() retournera la valeur de l'argument *expression testée*. Sinon elle retournera la valeur de l'argument *expression de remplacement*.

La fonction *COALESCE* retourne une valeur numérique ou une chaîne de caractères.

- *SELECT* salaire+commission *FROM* *employees*;
  - Posera un problème pour les employés qui n'ont de commission, il faut solutionner ce problème avec l'utilisation des fonctions de gestion de valeur *null*
- Solution : *SELECT* salaire+*COALESCE*(commission, 0) *FROM* *employees*;



## La gestion des valeurs *null* : fonctions natives

La gestion du *null* sous *SQL Server*

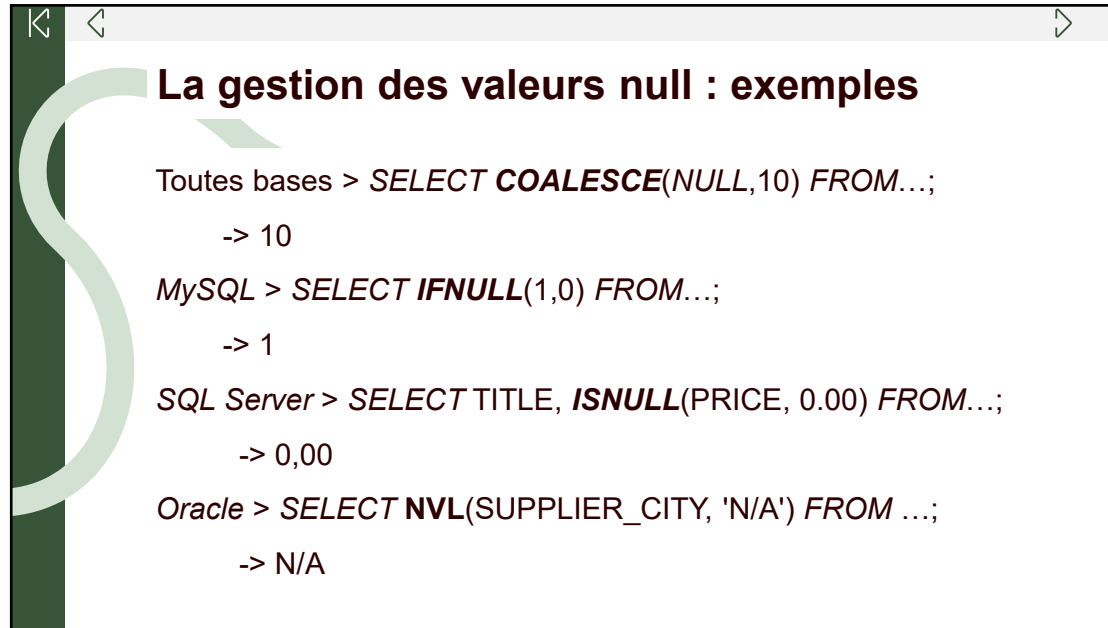
***ISNULL***(expression testée, expression de remplacement)

La gestion du *null* sous *Oracle*

***NVL***(expression testée, expression de remplacement)

La gestion du *null* sous *MySQL*

***IFNULL***(expression testée, expression de remplacement)



## La gestion des valeurs null : exemples

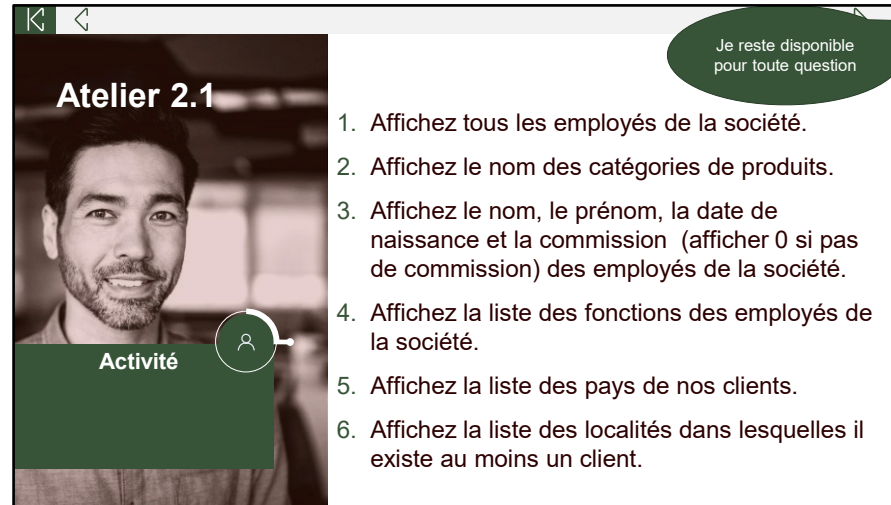
Toutes bases > *SELECT* **COALESCE**(NULL,10) *FROM*...;  
-> 10

MySQL > *SELECT* **IFNULL**(1,0) *FROM*...;  
-> 1

SQL Server > *SELECT* TITLE, **ISNULL**(PRICE, 0.00) *FROM*...;  
-> 0,00

Oracle > *SELECT* **NVL**(SUPPLIER\_CITY, 'N/A') *FROM* ...;  
-> N/A





**Atelier 2.1**

**Activité**

Je reste disponible pour toute question

1. Affichez tous les employés de la société.
2. Affichez le nom des catégories de produits.
3. Affichez le nom, le prénom, la date de naissance et la commission (afficher 0 si pas de commission) des employés de la société.
4. Affichez la liste des fonctions des employés de la société.
5. Affichez la liste des pays de nos clients.
6. Affichez la liste des localités dans lesquelles il existe au moins un client.

Atelier  
2.1...suite

Activité


7. Affichez les produits commercialisés et la valeur de stock par produit ( prix unitaire \* unités en stock).

8. Affichez le nom, le prénom, l'âge et l'ancienneté des employés.

9. Écrivez la requête qui permet d'afficher :

| Employé | a un  | gain annuel | sur 12 mois |
|---------|-------|-------------|-------------|
| Fuller  | gagne | salaire*12  | par an.     |
| Daviolo | gagne | salaire*12  | par an.     |

Je reste disponible pour toute question



Ce qu'il faut retenir

Echanges

- Respecter l'ordre d'introduction des clauses : ***SELECT...FROM***
- Le **select** peut contenir trois types d'expression : une **colonne**, une **constante**, un **calcul**
- Penser à la problématique des valeurs *null*



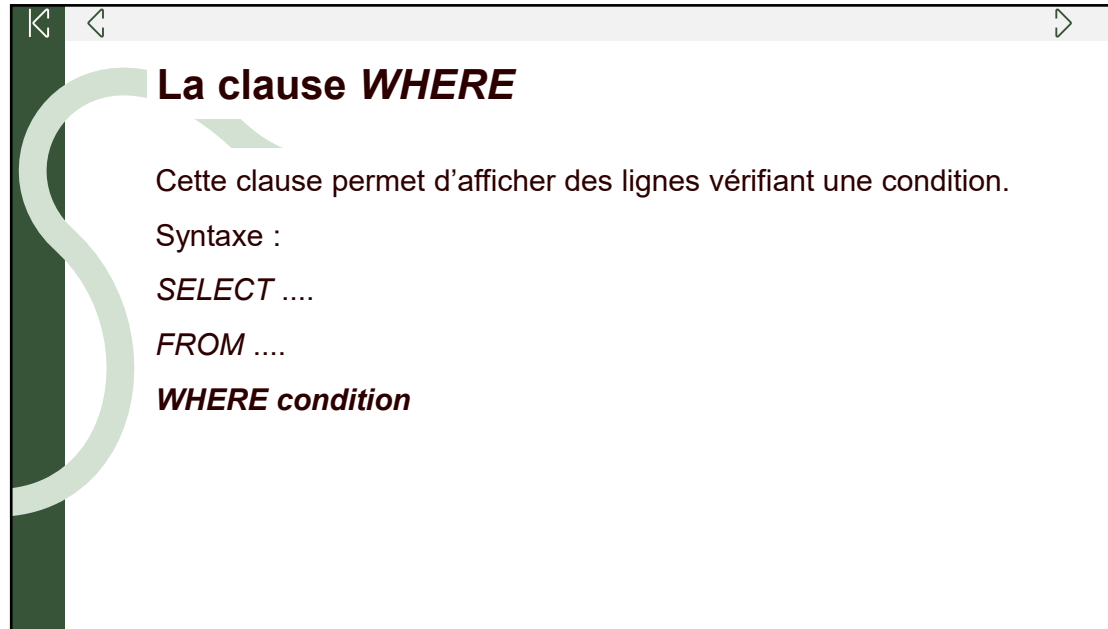
**Le Langage d'Interrogation des Données (LID) : les bases**

- La Sélection de données
- Gestion des valeurs *null*
- Les restrictions ou conditions
- Les tris
- Les jointures

A large green bracket is positioned to the left of the last three items in the list.

### Plan de la formation

| Module 1 - Introduction           | Module 2 – Le LID                | Module 3 – Fonctions de calcul   | Module 5 – Sous-interrogations  |
|-----------------------------------|----------------------------------|--|---------------------------------|
| Rappels sur le modèle relationnel | La sélection des données         | Les fonctions de calcul  | Les sous-requêtes dans le where |
| Schéma de la BDD du stage         | Atelier                          | Atelier – fonctions de regroupement  | Les sous-requêtes dans le from  |
| Caractéristiques du langage SQL   | Les conditions (ou restrictions) | Objectifs : <ul style="list-style-type: none"><li>• Ne ramener que les lignes qui nous intéressent</li></ul> |                                 |
|                                   | Atelier                          |  |                                 |
|                                   | Les tris                         |  |                                 |
|                                   | Atelier                          | Module 4 – Opérateurs ensemblistes   | Module 6 – Le LMD               |
|                                   | Les jointures entre tables       | Les opérateurs ensemblistes  | Modifier les données            |
|                                   | Atelier                          | Atelier  | Atelier                         |
|                                   |                                  |  | Module 7 – Notions sur le LDD   |

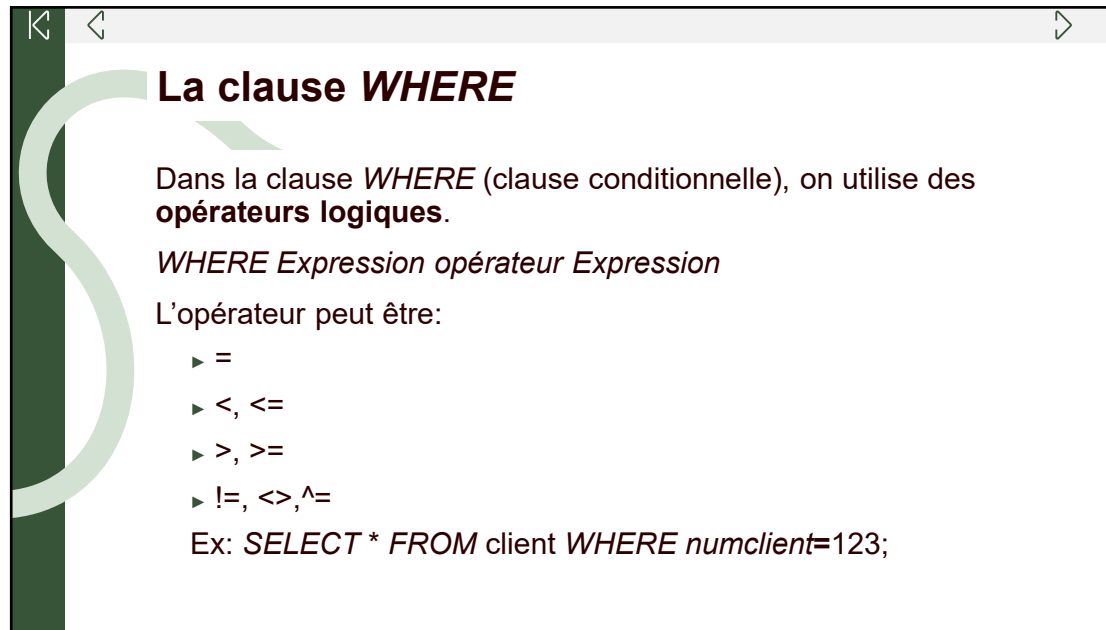


## La clause *WHERE*

Cette clause permet d'afficher des lignes vérifiant une condition.

Syntaxe :

```
SELECT ....  
FROM ....  
WHERE condition
```



## La clause *WHERE*

Dans la clause *WHERE* (clause conditionnelle), on utilise des **opérateurs logiques**.

*WHERE Expression opérateur Expression*

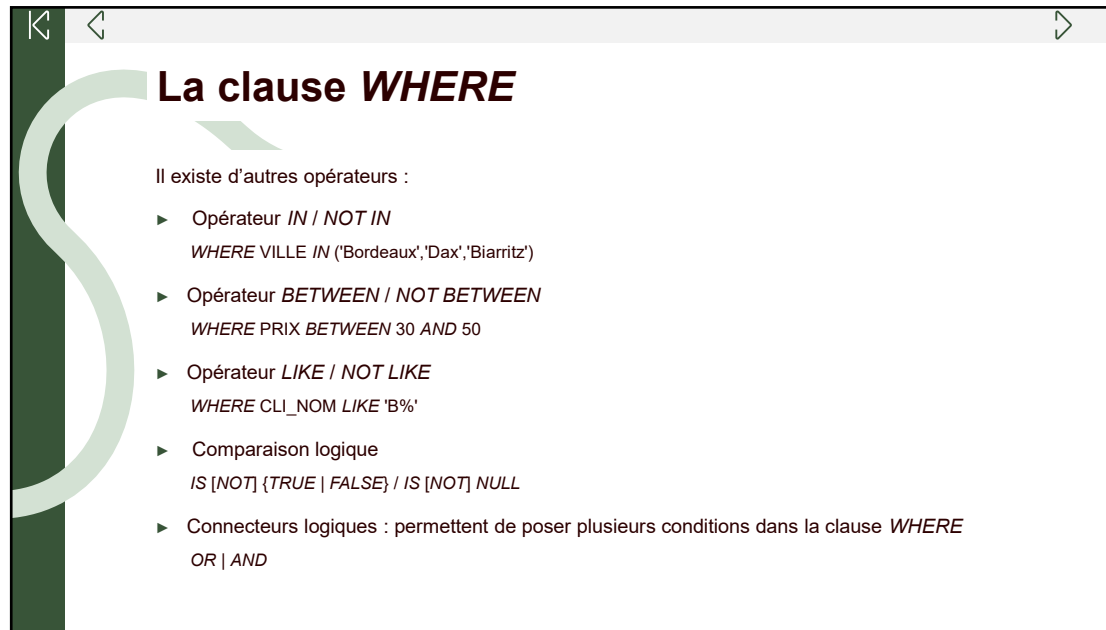
L'opérateur peut être:

- ▶ =
- ▶ <, <=
- ▶ >, >=
- ▶ !=, <>, ^=

Ex: *SELECT \* FROM client WHERE numclient=123;*

- = Opérateur d'égalité entre 2 valeurs
- < Strictement inférieure à
- <= Inférieure ou égale
- > Strictement supérieure à
- >= Supérieure ou égale
- !=, <>, ^= Différent de
  - ^= ne marche pas sous SQL Server et MySQL



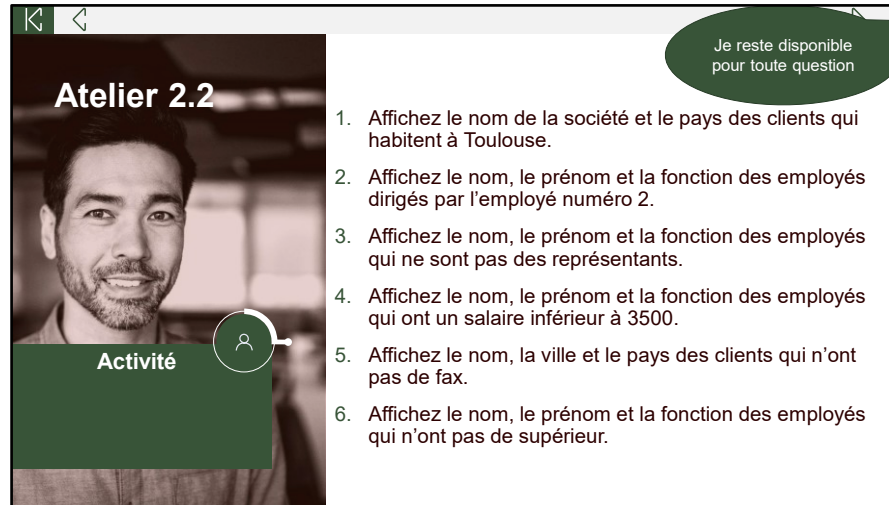


## La clause *WHERE*

Il existe d'autres opérateurs :

- ▶ Opérateur *IN* / *NOT IN*  
`WHERE VILLE IN ('Bordeaux','Dax','Biarritz')`
- ▶ Opérateur *BETWEEN* / *NOT BETWEEN*  
`WHERE PRIX BETWEEN 30 AND 50`
- ▶ Opérateur *LIKE* / *NOT LIKE*  
`WHERE CLI_NOM LIKE 'B%'`
- ▶ Comparaison logique  
`IS [NOT] {TRUE | FALSE} / IS [NOT] NULL`
- ▶ Connecteurs logiques : permettent de poser plusieurs conditions dans la clause *WHERE*  
`OR` | `AND`

- Explication de la syntaxe :
  - | veut dire ou
  - [ ] veut dire possible et facultatif
  - { } veut dire item à choisir
- *WHERE CLI\_NOM LIKE 'B\_'* = un caractère quelconque
- *BETWEEN 3000 AND 2000* cad  $\geq 3000$  et  $\leq 2000 \rightarrow$  pas de résultat
- *BETWEEN* marche dans *Oracle* et *PostGre* avec des lettres (mais exclusif sur le  $\leq$ )
- *AND* est prioritaire sur l'opérateur *OR*.
- On utilisera des jeux de parenthèses pour imposer une priorité.
  - Exemple : .... *WHERE (... OR ...) AND (... OR...)*
- Attention au piège des valeur *NULL* :
  - *SELECT* nom, commission *FROM* employes *WHERE* commission<500 *OR* commission *IS NULL*;
  - Si on n'ajoute pas la 2<sup>ème</sup> condition, les valeurs *NULL* ne seront pas dans le résultat




**Atelier 2.2**

**Activité**

Je reste disponible pour toute question

1. Affichez le nom de la société et le pays des clients qui habitent à Toulouse.
2. Affichez le nom, le prénom et la fonction des employés dirigés par l'employé numéro 2.
3. Affichez le nom, le prénom et la fonction des employés qui ne sont pas des représentants.
4. Affichez le nom, le prénom et la fonction des employés qui ont un salaire inférieur à 3500.
5. Affichez le nom, la ville et le pays des clients qui n'ont pas de fax.
6. Affichez le nom, le prénom et la fonction des employés qui n'ont pas de supérieur.



**Ce qu'il faut retenir**

Echanges

- Mettre les valeurs de type texte et date entre simples *quotes* (apostrophes)
- Chez certains éditeurs de *BDD*, le *WHERE* est sensible à la casse des valeurs
- Si on utilise, dans le même *WHERE*, à la fois plusieurs *or* et *and* ➔ *and* est prioritaire. Utiliser des parenthèses pour forcer la priorité.

**Le Langage d'Interrogation des Données (LID) : les bases**

- La Sélection de données
- Gestion des valeurs *null*
- Les restrictions ou conditions
- Les tris
- Les jointures

A large green bracket is positioned to the left of the last three items in the list, grouping them together.

### Plan de la formation

| Module 1 - Introduction           | Module 2 – Le LID                | Module 3 – Fonctions de calcul                                    | Module 5 – Sous-interrogations  |
|-----------------------------------|----------------------------------|---|---------------------------------|
| Rappels sur le modèle relationnel | La sélection des données         | Les fonctions de calcul   | Les sous-requêtes dans le where |
| Schéma de la BDD du stage         | Atelier                          | Atelier : fonctions de regroupement                               | Les sous-requêtes dans le from  |
| Caractéristiques du langage SQL   | Les conditions (ou restrictions) | Les clauses Group by et Having                                    | Atelier                         |
|                                   | Atelier                          | Atelier   |                                 |
|                                   | Les tris                         | Objectifs :<br>• Récupérer les informations dans l'ordre souhaité |                                 |
|                                   | Atelier                          |   |                                 |
|                                   | Les jointures entre tables       | Module 4 – Opérateurs ensemblistes                                | Module 6 – Le LMD               |
|                                   | Atelier                          | Les opérateurs ensemblistes                                       | Modifier les données            |
|                                   |                                  | Atelier   | Atelier                         |
|                                   |                                  |   | Module 7 – Notions sur le LDD   |




## La clause ORDER BY

Cette clause permet de définir le tri des colonnes.

**ASC** spécifie l'ordre **ascendant** et **DESC** l'ordre **descendant** du tri. ASC ou DESC peut être omis. Dans ce cas c'est l'ordre **ascendant** qui est utilisé **par défaut**.

```
SELECT ...  
FROM ...  
WHERE ...  
ORDER BY nom_colonne;  
  
SELECT ...  
FROM ...  
WHERE ...  
ORDER BY nom_colonne DESC;
```

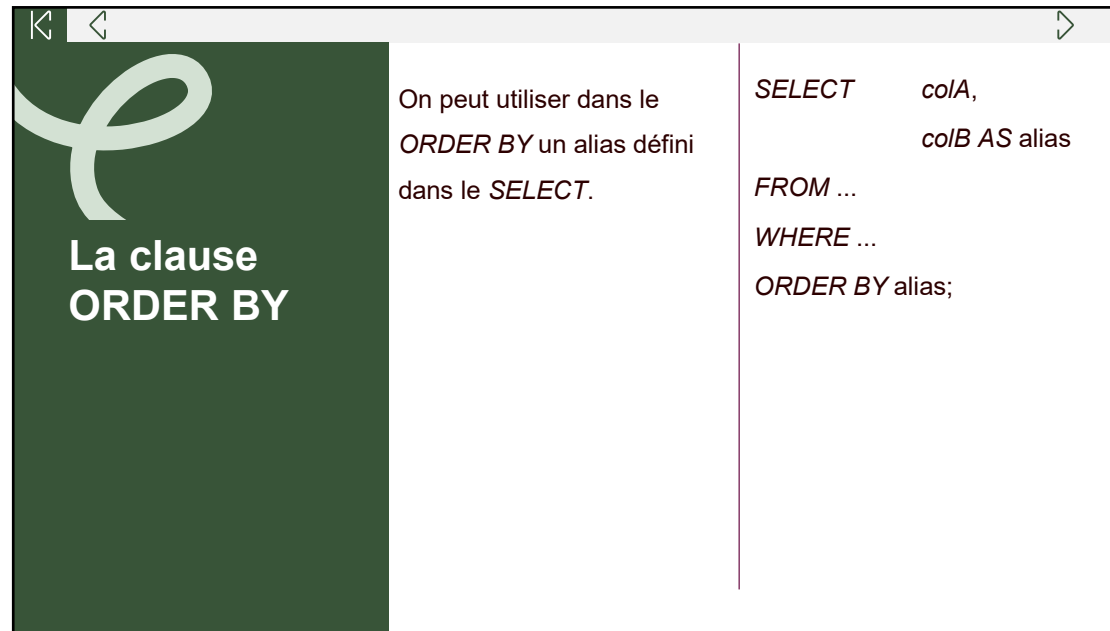


## La clause ORDER BY

Il est possible de trier, successivement, sur plusieurs colonnes.

On peut dans cette clause faire référence à la position de cette colonne dans le *SELECT*. Ici on trie d'abord sur la colonne **B** **en décroissant**, puis sur la **colonne A de façon croissante**.

```
SELECT ...  
FROM ...  
WHERE ...  
ORDER BY colA, colB;  
  
SELECT colA, colB  
FROM ....  
WHERE ....  
ORDER BY 2 DESC, 1;
```



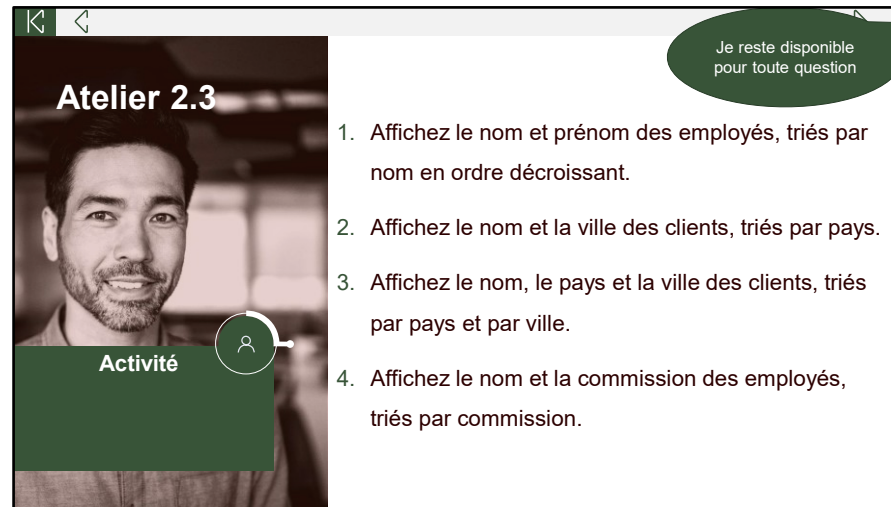
The slide is titled "La clause ORDER BY" and features a green background on the left with a white logo. The main content area is white and contains text explaining the use of the ORDER BY clause. A SQL query example is shown on the right side of the slide.

**La clause ORDER BY**

On peut utiliser dans le *ORDER BY* un alias défini dans le *SELECT*.

```
SELECT    colA,  
          colB AS alias  
FROM ...  
WHERE ...  
ORDER BY alias;
```






**Atelier 2.3**

Je reste disponible pour toute question

**Activité**

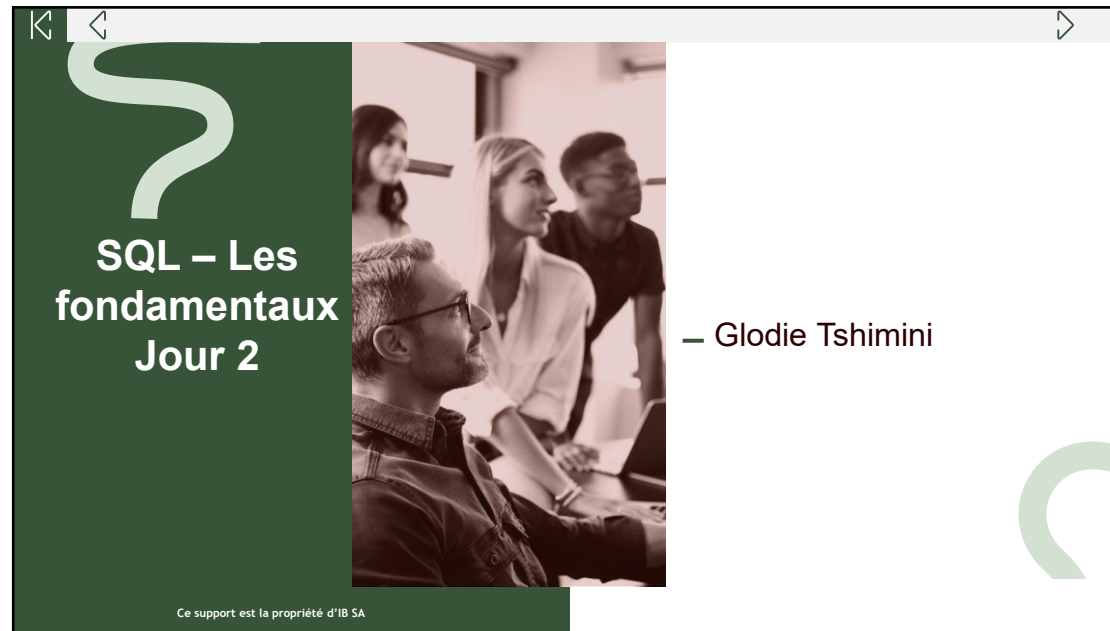
1. Affichez le nom et prénom des employés, triés par nom en ordre décroissant.
2. Affichez le nom et la ville des clients, triés par pays.
3. Affichez le nom, le pays et la ville des clients, triés par pays et par ville.
4. Affichez le nom et la commission des employés, triés par commission.



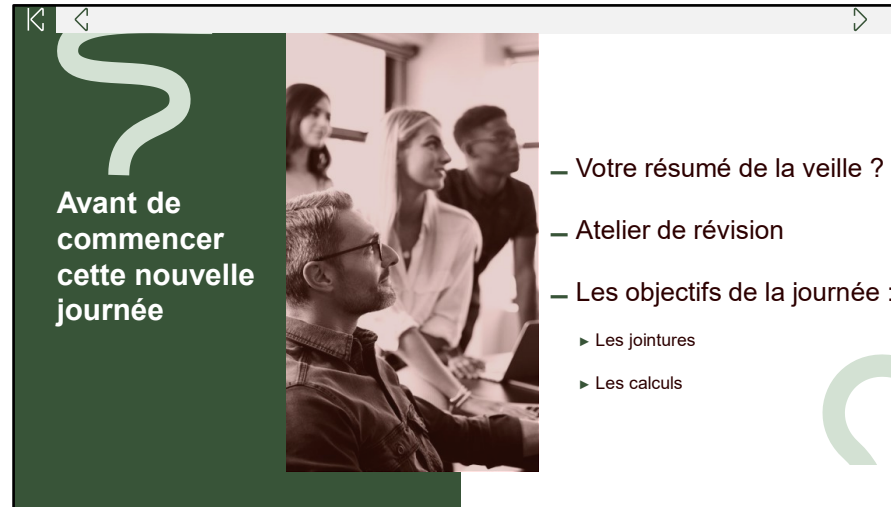
**Ce qu'il faut retenir**

Echanges

- La clause *ORDER BY* se saisit toujours à la fin de la requête
- Clause qui impacte de façon conséquente le temps de réponse des requêtes







Avant de commencer cette nouvelle journée

- Votre résumé de la veille ?
- Atelier de révision
- Les objectifs de la journée :
  - ▶ Les jointures
  - ▶ Les calculs

Cf. "Révisions jour 1"

**Le Langage d'Interrogation des Données (LID) : les bases**

- La Sélection de données
- Gestion des valeurs *null*
- Les restrictions ou conditions
- Les tris
- Les jointures

A large green bracket is positioned to the left of the last three items in the list, grouping them together.

### Plan de la formation

| Module 1 - Introduction           | Module 2 – Le LID                | Module 3 – Fonctions de calcul      | Module 5 – Sous-interrogations  |
|-----------------------------------|----------------------------------|-------------------------------------|---------------------------------|
| Rappels sur le modèle relationnel | La sélection des données         | Les fonctions de calcul             | Les sous-requêtes dans le where |
| Schéma de la BDD du stage         | Atelier                          | Atelier : fonctions de regroupement | Les sous-requêtes dans le from  |
| Caractéristiques du langage SQL   | Les conditions (ou restrictions) | Les clauses Group by et Having      | Atelier                         |
|                                   | Atelier                          | Atelier                             |                                 |
|                                   | Les tris                         |                                     |                                 |
|                                   | Atelier                          | Module 4 – Opérateurs               | Module 6 – Le LMD               |
|                                   | Les jointures entre tables       |                                     | données                         |
|                                   | Atelier                          | Atelier                             | Atelier                         |
|                                   |                                  |                                     |                                 |
|                                   |                                  |                                     | Module 7 – Notions sur le LDD   |

Objectifs :  
• Travailler avec des informations  
contenues dans différentes tables

## Jointures dans le *WHERE*

Jointures entre deux tables.

Soit deux tables : *table1* et *table2*. *table1* a les colonnes *col1* et *col2* et *table2* les colonnes *cola*, *colb*.

Supposons que le contenu des tables soit le suivant :

| table1 | col1 | col2 |
|--------|------|------|
|        | x    | 3    |
|        | y    | 4    |

| table2 | cola | colb |
|--------|------|------|
|        | a    | 7    |
|        | b    | 4    |

Soit la commande :

```
SELECT col1, cola FROM table1, table2
WHERE table1.col2=table2.colb;
```

Cette requête extrait des données de deux tables : *table1* et *table2* avec une condition entre deux colonnes de tables différentes.

- Exemple pratique : affichez le numéro de commande avec le nom de l'employé et du client
  - Ici jointure entre les tables *employes*, *commandes* et *clients* pour obtenir toutes ses informations

```
SELECT e.prenom AS prenom_employe,
       e.nom AS nom_employe,
       c.societe AS client,
       o.no_commande AS commande
FROM clients c, employes e, commandes o
WHERE c.code_client = o.code_client
AND e.no_employe = o.no_employe
```



## Jointures dans le *WHERE*

### Comment fonctionne-t-elle ?

Une nouvelle table est construite avec pour colonnes l'ensemble des colonnes des deux tables et pour lignes le produit cartésien des deux tables :

| col1 | col2 | cola | colb |
|------|------|------|------|
| x    | 3    | a    | 7    |
| x    | 3    | b    | 4    |
| y    | 4    | a    | 7    |
| y    | 4    | b    | 4    |

La condition *WHERE col2=colb* est appliquée à cette nouvelle table. On obtient donc la nouvelle table suivante :

| col1 | col2 | cola | colb |
|------|------|------|------|
| y    | 4    | b    | 4    |

# Jointures

Il y a ensuite affichage des colonnes demandées (select) :

| coll | cola |
|------|------|
| y    | b    |



## Jointures dans le *WHERE*

Syntaxe d'une requête multi-tables :

***SELECT*** *colonne1, colonne2, ...*

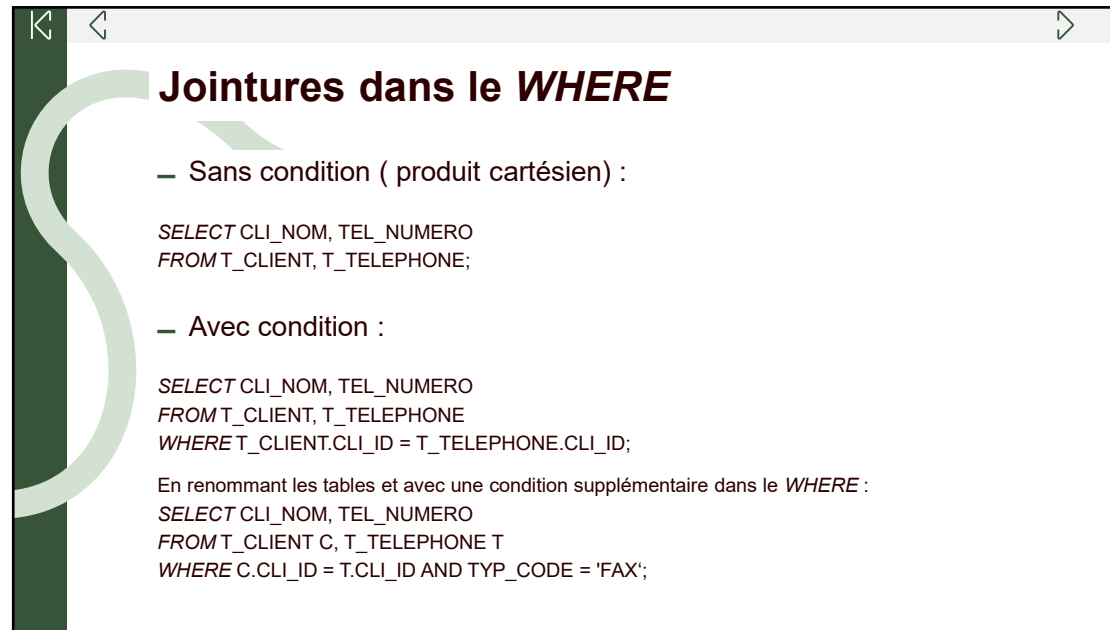
***FROM*** *table1, table2, ..., tablep*

***WHERE*** *jointure1*

***AND*** *jointure2*

***AND ...***

***ORDER BY ...;***



## Jointures dans le *WHERE*

- Sans condition ( produit cartésien ) :  

```
SELECT CLI_NOM, TEL_NUMERO  
FROM T_CLIENT, T_TELEPHONE;
```
- Avec condition :  

```
SELECT CLI_NOM, TEL_NUMERO  
FROM T_CLIENT, T_TELEPHONE  
WHERE T_CLIENT.CLI_ID = T_TELEPHONE.CLI_ID;
```

En renommant les tables et avec une condition supplémentaire dans le *WHERE* :

```
SELECT CLI_NOM, TEL_NUMERO  
FROM T_CLIENT C, T_TELEPHONE T  
WHERE C.CLI_ID = T.CLI_ID AND TYP_CODE = 'FAX';
```



## Jointures avec un *JOIN*

```
SELECT CLI_NOM, TEL_NUMERO  
FROM T_CLIENT NATURAL JOIN T_TELEPHONE;
```

La jointure se fait sur la colonne qui porte le même nom dans les deux tables. Et une seule colonne doit avoir le même nom de part et d'autre des tables

|                    |   |
|--------------------|---|
| Jointure naturelle | <pre>SELECT ...<br/>FROM &lt;table gauche&gt;<br/>    NATURAL JOIN &lt;table droite&gt;<br/>;</pre> |
|--------------------|---|

## Jointures avec un *JOIN*

```
SELECT CLI_NOM, TEL_NUMERO
FROM T_CLIENT JOIN T_TELEPHONE USING (CLI_ID);
```

La jointure se fait sur la colonne qui porte le même nom dans les deux tables.

|                          |   |
|--------------------------|---|
| Jointure<br>join...using | <pre>SELECT ... FROM &lt;table gauche&gt;       [INNER] JOIN &lt;table droite&gt;                 USING (nom de la colonne) ;</pre> |
|--------------------------|---|

## Jointures avec un *JOIN*

```
SELECT CLI_NOM, TEL_NUMERO
FROM T_CLIENT INNER JOIN T_TELEPHONE
ON T_CLIENT.CLI_ID = T_TELEPHONE.CLI_ID;
```

Ici on spécifie le nom des colonnes sur lesquelles se fait la jointure.

|                  |  |
|------------------|--|
| Jointure interne | <pre>SELECT ... FROM &lt;table gauche&gt; [INNER] JOIN &lt;table droite&gt; ON &lt;condition de jointure&gt; ;</pre> |
|------------------|--|



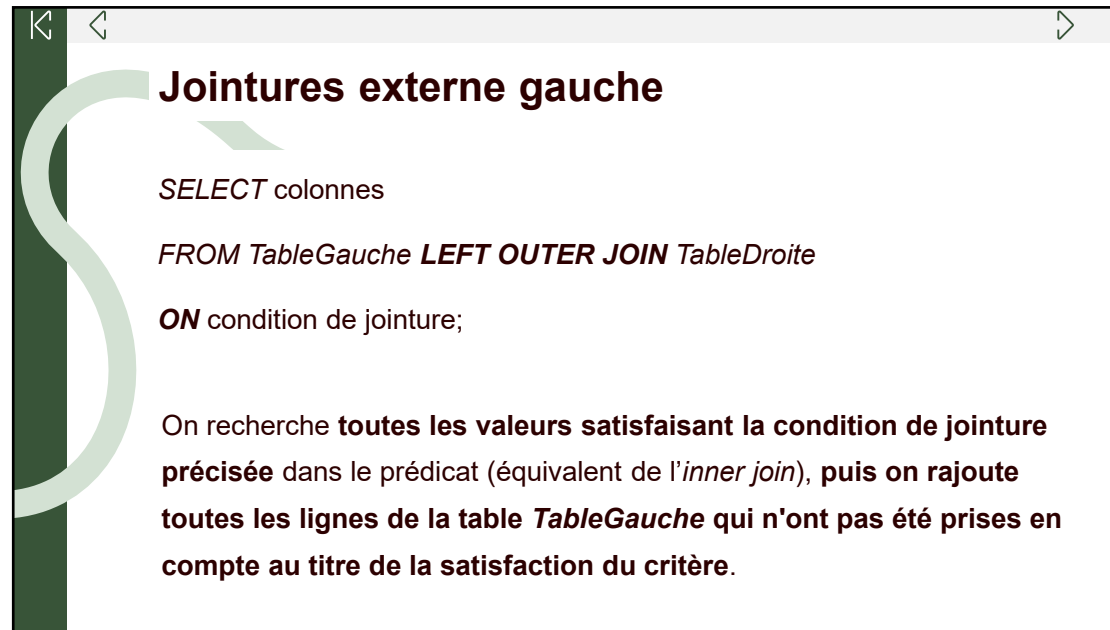
## Jointures avec un JOIN

```
SELECT CLI_NOM, TEL_NUMERO
FROM T_CLIENT C LEFT OUTER JOIN T_TELEPHONE T
ON C.CLI_ID = T.CLI_ID;
```

Les jointures externes rapatrient les informations disponibles, même si des lignes de la table ne sont pas renseignées entre les différentes tables jointes

|                  |  |
|------------------|--|
| Jointure externe | <pre>SELECT ... FROM &lt;table gauche&gt;     LEFT   RIGHT   FULL [OUTER] JOIN &lt;table droite&gt;     ON condition de jointure</pre> |
|------------------|--|

Par exemple, récupérer tous les produits même ceux qui n'ont pas de catégorie



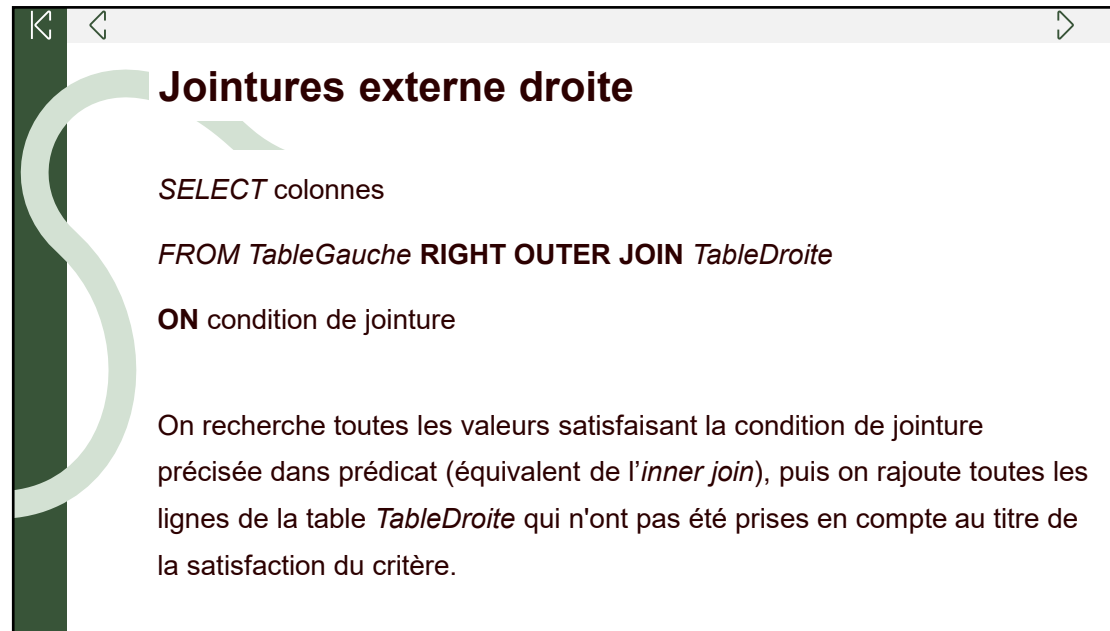
## Jointures externe gauche

*SELECT* colonnes

*FROM* *TableGauche* **LEFT OUTER JOIN** *TableDroite*

**ON** condition de jointure;

On recherche **toutes les valeurs satisfaisant la condition de jointure précisée** dans le prédicat (équivalent de l'*inner join*), **puis on rajoute toutes les lignes de la table *TableGauche* qui n'ont pas été prises en compte au titre de la satisfaction du critère.**



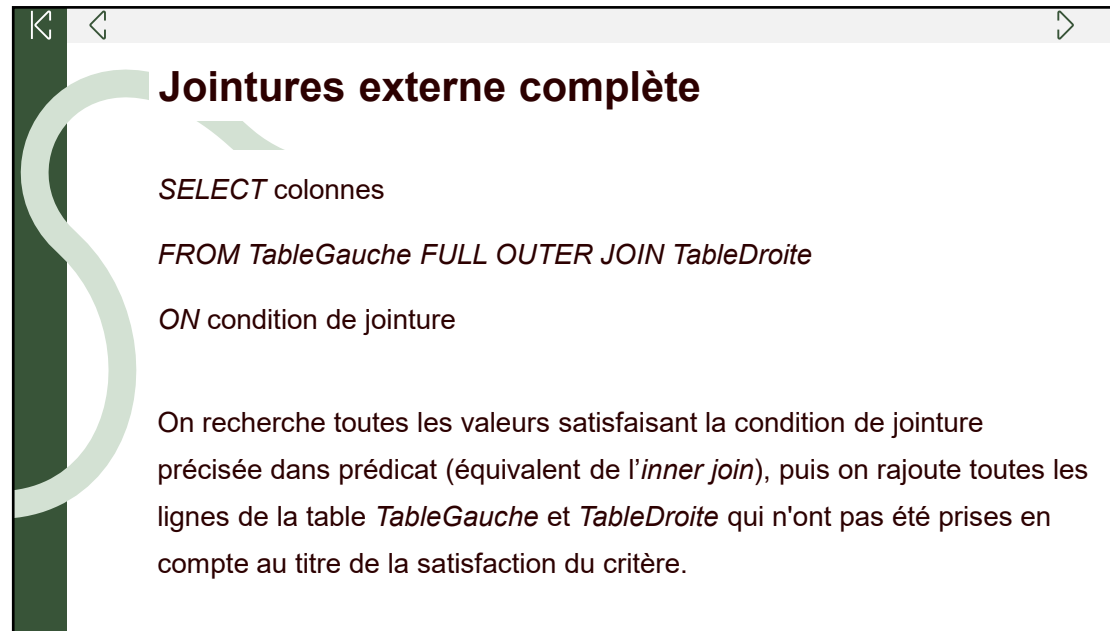
## Jointures externe droite

*SELECT* colonnes

*FROM* *TableGauche* **RIGHT OUTER JOIN** *TableDroite*

**ON** condition de jointure

On recherche toutes les valeurs satisfaisant la condition de jointure précisée dans prédicat (équivalent de l'*inner join*), puis on rajoute toutes les lignes de la table *TableDroite* qui n'ont pas été prises en compte au titre de la satisfaction du critère.



## Jointures externe complète

*SELECT* colonnes

*FROM TableGauche FULL OUTER JOIN TableDroite*

*ON* condition de jointure

On recherche toutes les valeurs satisfaisant la condition de jointure précisée dans prédicat (équivalent de l'*inner join*), puis on rajoute toutes les lignes de la table *TableGauche* et *TableDroite* qui n'ont pas été prises en compte au titre de la satisfaction du critère.

## Jointures croisées

```
SELECT CHB_ID, TRF_DATE_DEBUT, 0 AS TRF_CHB_PRIX
FROM T_TARIF CROSS JOIN T_CHAMBRE
ORDER BY CHB_ID, TRF_DATE_DEBUT;
```

On fait sciemment le **produit cartésien** .

|                  |   |
|------------------|---|
| Jointure croisée | <pre>SELECT ... FROM &lt;table gauche&gt;     CROSS JOIN &lt;table droite&gt;</pre> |
|------------------|---|

Atelier 2.4

Activité

Je reste disponible pour toute question

1. Affichez le nom, le prénom, la fonction et le salaire des employés qui ont un salaire compris entre 2500 et 3500.

2. Affichez le nom du produit, le nom du fournisseur, le nom de la catégorie et les quantités de produits qui ne sont pas d'une des catégories 1,3,5 et 7.

3. Affichez le nom du produit, le nom du fournisseur, le nom de la catégorie et les quantités des produits qui ont un numéro de fournisseur entre 1 et 3 ou un code catégorie entre 1 et 3, et pour lesquelles les quantités sont données en boîte(s) ou en carton(s).




The screenshot shows a presentation slide with a dark background on the left and a white background on the right. The left side features a portrait of a man and the text 'Atelier 2.4...suite' and 'Activité'. The right side contains a list of three SQL exercises. A green speech bubble in the top right corner says 'Je reste disponible pour toute question'.

### Atelier 2.4...suite

Activité

4. Écrivez la requête qui permet d'afficher le nom des employés qui ont effectué au moins une vente pour un client parisien.
5. Affichez le nom des produits et le nom des fournisseurs pour les produits des catégories 1,4 et 7.
6. Affichez le nom des employés ainsi que le nom de leur supérieur hiérarchique

Je reste disponible pour toute question

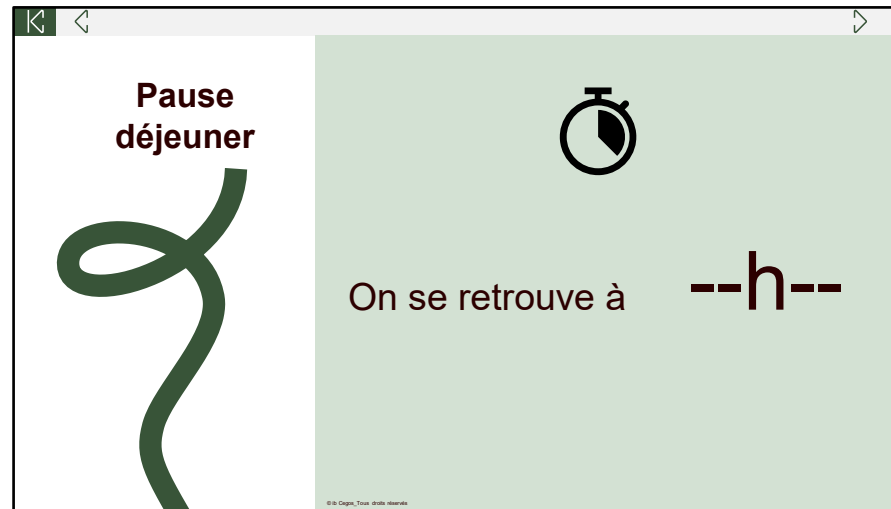


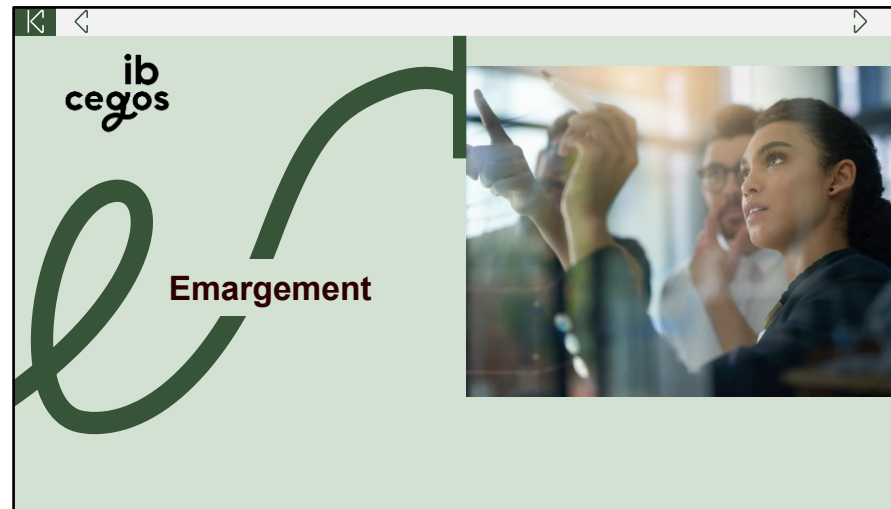
Ce qu'il faut retenir

Echanges

- Sans jointures, on obtient le produit cartésien des tables (résultat incohérent).
- Le **JOIN** est plus utilisé que le **WHERE**.
- Les 2 colonnes communes effectuant la jointure entre 2 tables doivent avoir le même type de données. Pas forcément le même nom.





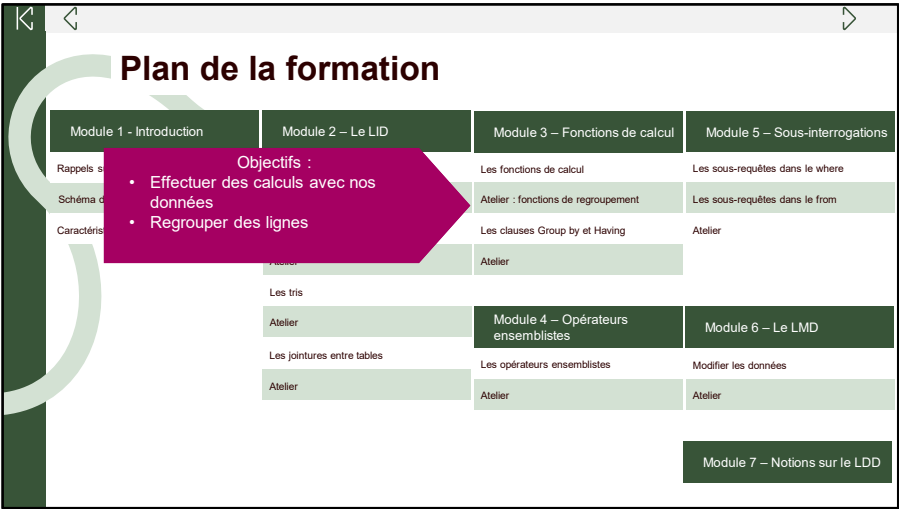


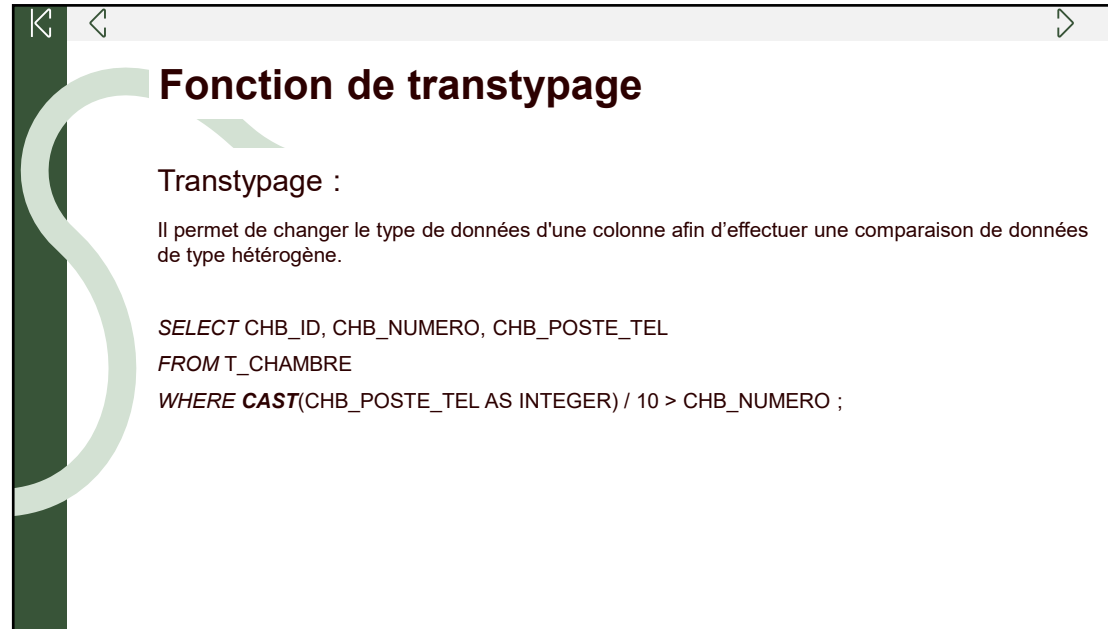
Émargez **chaque début de demi-journée avec une vraie signature** en scannant le **QR Code** projeté dans la salle ou via le lien reçu par votre email d'inscription à cette formation.



Pour ce chapitre, les ateliers ne porteront que sur les fonctions de regroupement.  
Vous pouvez explorer toutes les fonctions à travers les documentations de chaque SGBDR

- SQL Server [https://www.w3schools.com/sql/sql\\_ref\\_sqlserver.asp](https://www.w3schools.com/sql/sql_ref_sqlserver.asp)
- MySQL [https://www.w3schools.com/sql/sql\\_ref\\_mysql.asp](https://www.w3schools.com/sql/sql_ref_mysql.asp)
- PostgreSQL <https://www.postgresql.org/docs/current/functions.html>
- Oracle <https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Functions.html>





## Fonction de transtypage

Transtypage :

Il permet de changer le type de données d'une colonne afin d'effectuer une comparaison de données de type hétérogène.

```
SELECT CHB_ID, CHB_NUMERO, CHB_POSTE_TEL
FROM T_CHAMBRE
WHERE CAST(CHB_POSTE_TEL AS INTEGER) / 10 > CHB_NUMERO ;
```

Pour changer de type la fonction **CAST()** est universel cad elle est disponible pour tous les SGBDR

- Oracle et Postgre
  - TO\_CHAR(valeur)
  - TO\_NUMBER(valeur)
  - TO\_DATE(string,'format') ex de format 'yymmdd'
- SQL Server
  - STR(champ)
  - STR\_TO\_DATE(string, "format")
  - CONVERT()
- MySQL
  - CONVERT()

## Fonctions de transtypage

|   |  |
|---|--|
| <i>TO_CHAR</i> (expression,'format')<br>(Oracle et Postgre) | date sous forme littérale                            |
| <i>TO_DATE</i> (expression,'format')<br>(Oracle et Postgre) | Convertit une chaîne de caractère en date            |
| <i>CONVERT</i> (type, expression)<br>(Sql Server)           | Convertit une chaîne de caractère au format souhaité |
| <i>CAST</i> ( expression as format)                         | Convertit une chaîne de caractère au format souhaité |

## Fonctions de chaînes de caractères

Mise en majuscule / Minuscule :

Les opérateurs *LOWER* et *UPPER* permettent de mettre en majuscule ou en minuscule des chaînes de caractères dans les requêtes.

```
SELECT upper(CLI_PRENOM), lower(CLI_NOM)
  T_CLIENT;
```

FROM

| CLI_PRENOM | CLI_NOM |
|------------|---------|
| ALAIN      | dupont  |
| MARC       | martin  |
| ALAIN      | bouvier |
| .....      |         |

- `SELECT upper(societe) FROM clients;`
- `SELECT societe FROM clients WHERE upper(ville)='PARIS';`
  - Attention aux chutes de performance en cas d'index sur la colonne ville

## Fonctions de chaînes de caractères

Extraire une sous chaîne :

La fonction *SUBSTRING* (*SUBSTR* sous Oracle et Postgre) permet d'extraire une sous chaîne d'une chaîne de caractère.

Exemple SQL Server :

```
SELECT CLI_NOM, CLI_PRENOM, SUBSTRING(CLI_PRENOM,1,1) + SUBSTRING(CLI_NOM,1,1) AS INITIALES FROM T_CLIENT;
```

| CLI_NOM | CLI_PRENOM | INITIALES |
|---------|------------|-----------|
| -----   | -----      | -----     |
| DUPONT  | Alain      | AD        |
| MARTIN  | Marc       | MM        |
| .....   |            |           |

- Création d’initiales à partir du prénom et nom de l’employé :
- **ORACLE et POSTGRE**
  - `SELECT nom, prenom, SUBSTR(prenom,1,1) || SUBSTR(nom,1,1) AS initiales FROM employes;`
- **MYSQL et SQL SERVER**
  - `SELECT nom, prenom, CONCAT(SUBSTRING(prenom,1,1), SUBSTRING(nom,1,1)) FROM employes;`



|  |  |
|--|--|
| Fonctions de chaînes de caractères   |  |
| <i>CONCAT</i> (expression1,expression2,...)  | concaténation : utiliser de préférence    chez Oracle (car deux arguments maximums avec la fonction) |
| <i>INITCAP</i> (expression)<br>(Oracle, Postgre)                                   | initiales en lettres capitales   |
| <i>LPAD</i> (expression,nb_caractères,caractère)<br>(Mysql ,Oracle, Postgre)       | complément ou troncature à n position à gauche   |
| <i>RPAD</i> (expression,nb_caractères,caractère)<br>(Mysql ,Oracle, Postgre)       | complément ou troncature à n position à droite   |
| <i>LTRIM</i> (expression)<br><i>RTRIM</i> (expression)<br><i>TRIM</i> (expression) | suppression des espaces en tête/queue d'une chaîne   |
| <i>REPLACE</i> (expression, valeur à remplacer, valeur de remplacement)            | remplacement   |
| <i>INSTR</i> (expression,'texte cherché')<br>(Mysql ,Oracle)                       | position d'une chaîne dans une sous chaîne   |
| <i>PATINDEX</i> (''%texte cherché%', expression)<br>(Sql Server)                   | position d'une chaîne dans une sous chaîne   |
| <i>POSITION</i> ('texte cherché' in expression)<br>(Postgre)                       | position d'une chaîne dans une sous chaîne   |

## Fonctions de chaînes de caractères

|  |   |
|--|---|
| <i>LENGTH</i> (expression)<br>(MySQL ,Oracle, Postgre) | longueur de la chaîne                       |
| <i>LEN</i> (expression)<br>( SQL Server)               | longueur de la chaîne                       |
| <i>ASCII</i> (expression)                              | code ASCII d'un caractère                   |
| <i>CHR</i> (valeur)<br>(Oracle et Postgre)             | caractère dont le code ASCII est donné      |
| <i>CHAR</i> (valeur)<br>(MySQL et SQL Server)          | caractère dont le code ASCII est donné      |
| <i>REVERSE</i> (expression)                            | Inverse l'ordre des caractères d'une chaîne |

## Fonctions date système

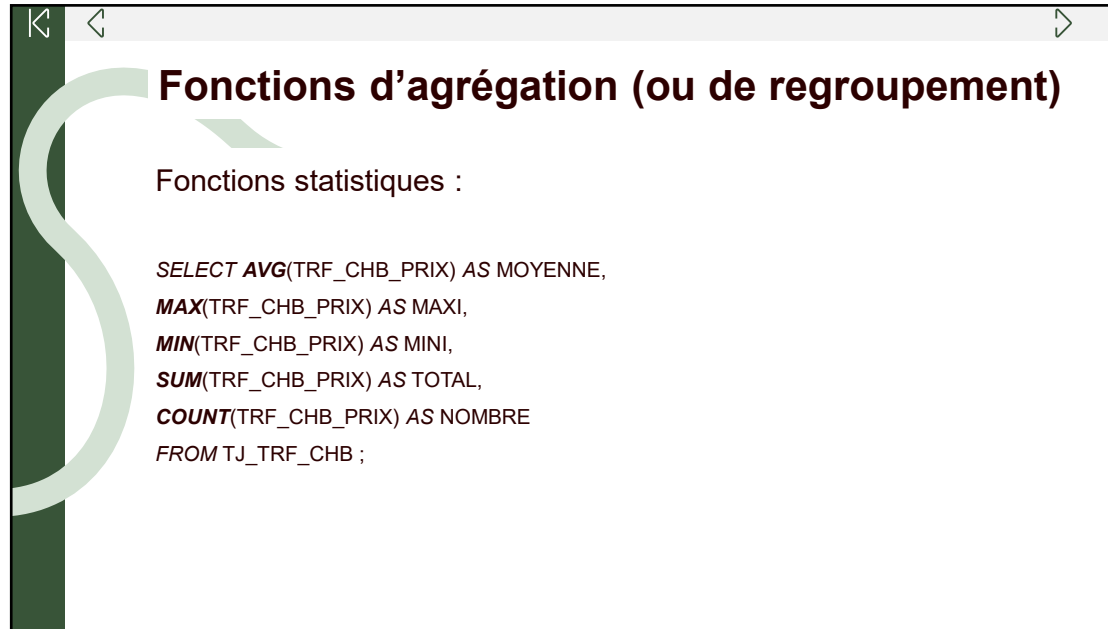
Heure et date courante :

Exemple Postgre, MySQL, Oracle :

```
SELECT NO_COMMANDE  
FROM COMMANDES WHERE  
DATE_ENVOI BETWEEN CURRENT_DATE  
and CURRENT_DATE - 14;
```

|                  |  |
|------------------|--|
| Oracle           | <b><i>SYSDATE<br/>CURRENT_DATE</i></b> |
| MySQL<br>Postgre | <b><i>CURRENT_DATE</i></b>             |
| SQL Server       | <b><i>GETDATE()</i></b>                |
| Access           | <b><i>DATE()</i></b>                   |

|   |  |
|---|--|
| Fonctions de dates  |  |
| Expression + <i>INTERVAL</i> '1' day (MySQL ,Oracle, Postgre)   | Ajoute des jours, des mois, des années à une date  |
| <i>DATEADD</i> (day,1,expression) (SQL Server)  | Ajoute des jours, des mois, des années à une date  |
| <i>ADD_MONTHS</i> (expression, nb_mois) (Oracle)  | Ajoute des mois à une date                         |
| <i>NEXT_DAY</i> (expression,'jour') (Oracle)  | Date du prochain jour d'un nom donné               |
| <i>LAST_DAY</i> (expression) (Oracle, MySQL)  | Renvoie le n° du dernier jour d'un mois d'une date |
| <i>MONTHS_BETWEEN</i> (date1,date2) (Oracle)  | Nombre de mois entre deux dates                    |
| <i>DATEDIFF</i> (day   month   year, date_debut,date_fin) (SQL Server)<br><i>DATEDIFF</i> (date_debut,date_fin) (MySQL) | Différence entre deux dates                        |
| <i>DATEPART</i> (day   month   year, expression) (SQL Server)   | Renvoie la valeur du jour, mois ou année           |
| <i>EXTRACT</i> (day   month   year from expression) (MySQL ,Oracle, Postgre)  | Renvoie la valeur du jour, mois ou année           |



**Fonctions d'agrégation (ou de regroupement)**

Fonctions statistiques :

```
SELECT AVG(TRF_CHB_PRIX) AS MOYENNE,  
MAX(TRF_CHB_PRIX) AS MAXI,  
MIN(TRF_CHB_PRIX) AS MINI,  
SUM(TRF_CHB_PRIX) AS TOTAL,  
COUNT(TRF_CHB_PRIX) AS NOMBRE  
FROM TJ_TRF_CHB ;
```

| Fonctions mathématiques   |                                 |
|---|---------------------------------|
| <i>ABS</i> (expression)   | Valeur absolue                  |
| <i>MOD</i> (expression)   | Modulo (reste division entière) |
| <i>SIGN</i> (expression)  | Signe                           |
| <i>SQRT</i> (expression)  | Racine carrée                   |
| <i>CEIL</i> (expression) ; SQL Server : <i>CEILING</i> (expression)                               | Plus grand entier               |
| <i>FLOOR</i> (expression)   | Plus petit entier               |
| <i>ROUND</i> (expression, nb_décimales)   | Arrondi                         |
| <i>TRUNC</i> (expression, nb_décimales)<br>SQL Server : <i>ROUND</i> (expression, nb_décimales,1) | Tronqué                         |
| <i>EXP</i> (expression)   | Exponentielle                   |
| <i>LN</i> (expression)  | Logarithme népérien             |
| <i>LOG</i> (expression)   | Logarithme décimal              |
| <i>POWER</i> (expression,valeur)  | Puissance                       |
| <i>COS</i> (expression) ; <i>SIN</i> ((expression)  | Cosinus ; Sinus                 |
| <i>TAN</i> (expression)   | Tangente                        |
| <i>PI</i> () ; Oracle : <i>ASIN</i> (1)*2   | Constante PI                    |

- Exemple la moyenne des unités en stock, arrondi à 2 décimales
  - `SELECT ROUND(AVG(unites_stock),2) AS moyenne FROM details_commandes;`
  - Ou `SELECT ROUND(AVG(COALESCE(unites_stock,0),2) AS moyenne FROM details_commandes;`

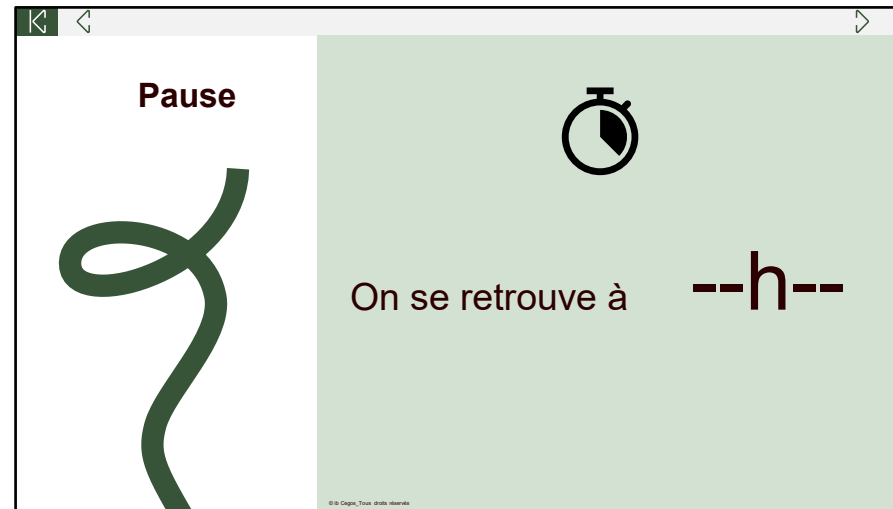


**Atelier 3.1**

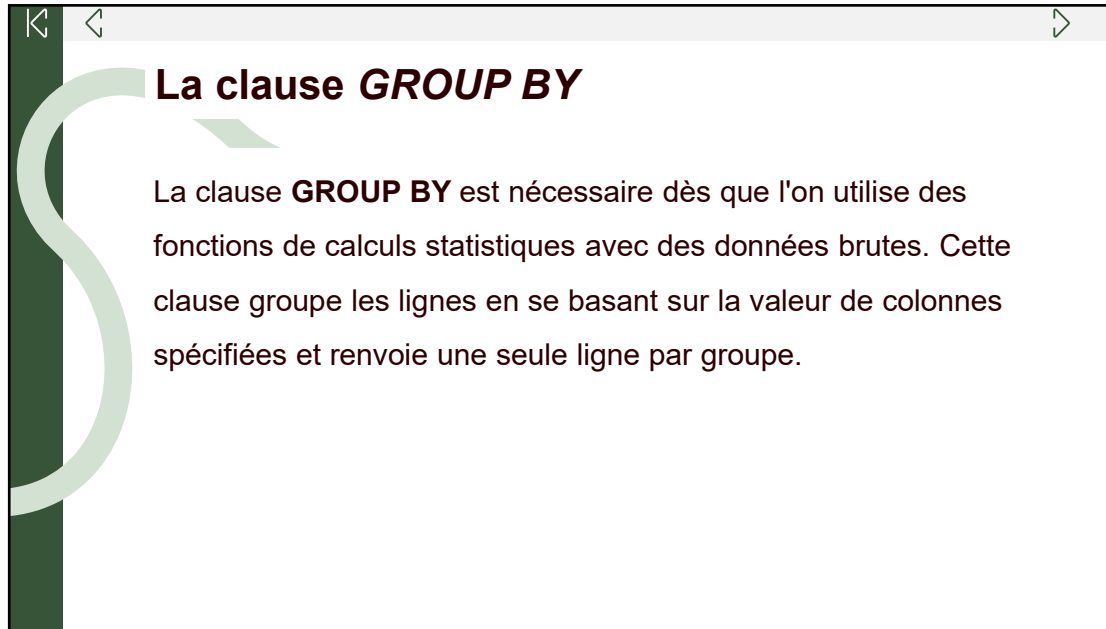
**Activité**

1. Affichez la somme des salaires et des commissions des employés (cumuler les 2 en une seule colonne).
2. Affichez la moyenne des salaires et la moyenne des commissions des employés.
3. Affichez le salaire maximum et la plus petite commission des employés.
4. Affichez le nombre distinct de fonction.

Je reste disponible pour toute question







## La clause **GROUP BY**

La clause **GROUP BY** est nécessaire dès que l'on utilise des fonctions de calculs statistiques avec des données brutes. Cette clause groupe les lignes en se basant sur la valeur de colonnes spécifiées et renvoie une seule ligne par groupe.

## La clause GROUP BY

Sans group by :

```
SELECT COUNT(CHB_ID) AS NOMBRE, CHB_ETAGE FROM T_CHAMBRE;
```

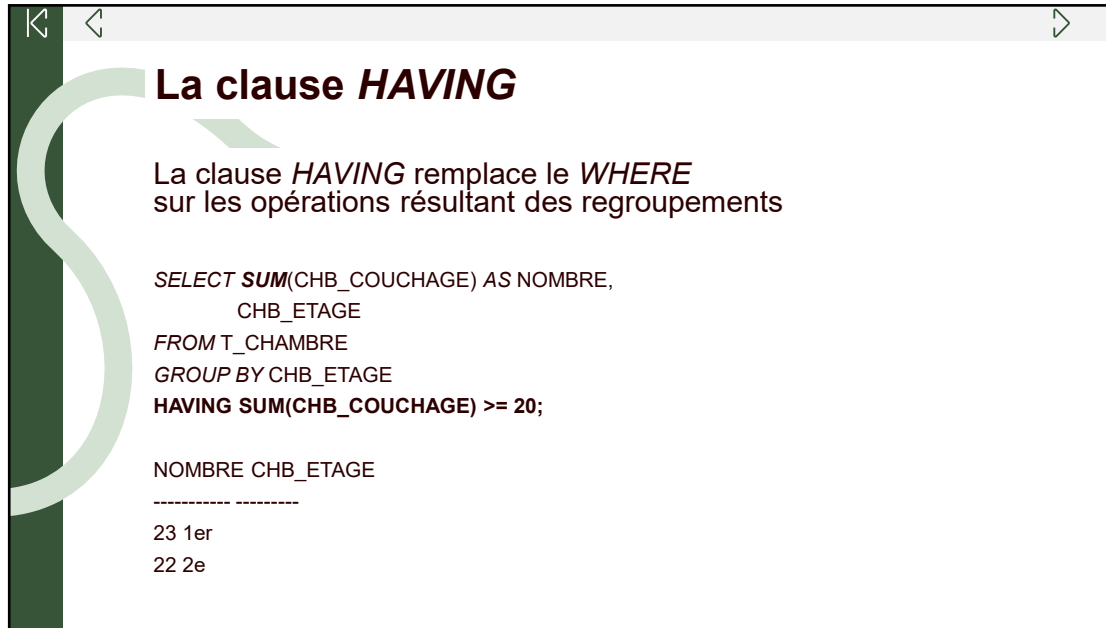
| NOMBRE | CHB_ETAGE |
|--------|-----------|
| 1      | RDC       |
| 1      | RDC       |
| 1      | RDC       |
| 1      | RDC       |
| 1      | 1er       |
| 1      | 1er       |
| .....  |           |

## La clause *GROUP BY*

Avec Group By :

```
SELECT COUNT(CHB_ID) AS NOMBRE, CHB_ETAGE FROM T_CHAMBRE
GROUP BY CHB_ETAGE;
```

| NOMBRE | CHB_ETAGE       |
|--------|-----------------|
| 8      | 1 <sup>er</sup> |
| 8      | 2 <sup>e</sup>  |
| 4      | RDC             |



## La clause *HAVING*

La clause *HAVING* remplace le *WHERE* sur les opérations résultant des regroupements

```
SELECT SUM(CHB_COUCHAGE) AS NOMBRE,  
       CHB_ETAGE  
FROM T_CHAMBRE  
GROUP BY CHB_ETAGE  
HAVING SUM(CHB_COUCHAGE) >= 20;
```

| NOMBRE | CHB_ETAGE |
|--------|-----------|
| 23     | 1er       |
| 22     | 2e        |




**Atelier 3.2**

Je reste disponible pour toute question

1. Écrivez la requête qui permet d'afficher la masse salariale des employés par fonction.
2. Affichez le montant de chaque commande, en ne conservant que les commandes qui comportent plus de 5 références de produit.
3. Afficher la valeur des produits en stock et la valeur des produits commandés par fournisseur, pour les fournisseurs qui ont un numéro compris entre 3 et 6.

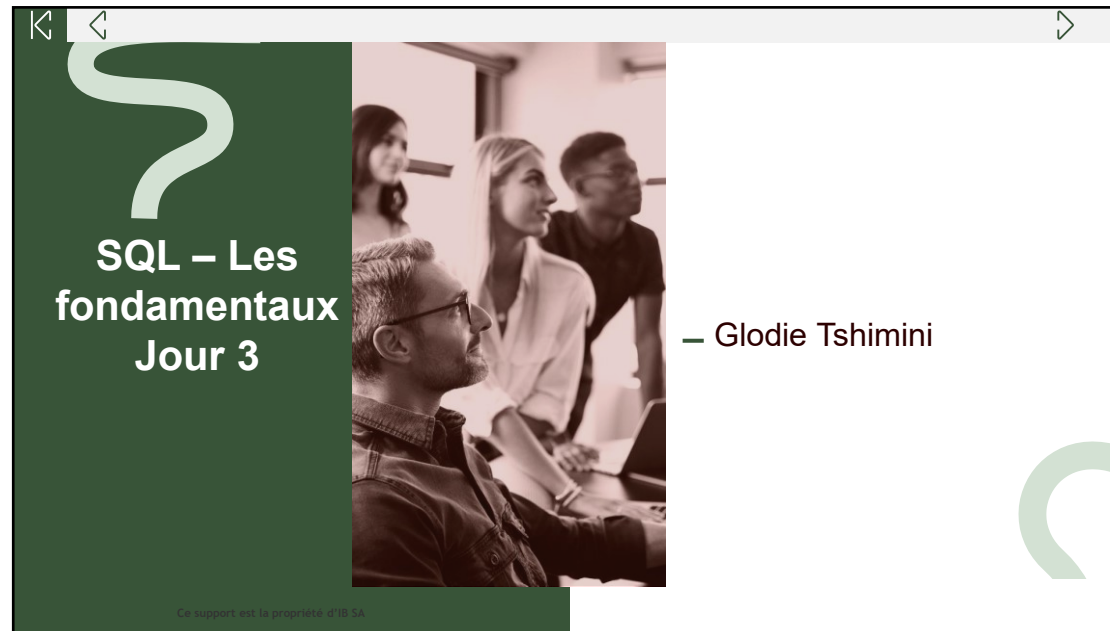
Activité



**Ce qu'il faut retenir**

Echanges

- Lorsque dans un *SELECT* on se retrouve avec au moins une fonction de regroupement et au moins une expression autre = tout ce qui n'est pas fonction de regroupement doit être dans la clause *GROUP BY*.
- La clause *HAVING* filtre en prenant en compte la clause *GROUP BY*.  
Alors que la clause *WHERE* intervient sur les lignes du *FROM* (et des éventuelles jointures).







**Avant de commencer cette nouvelle journée**

- Votre résumé de la veille ?
- Atelier de révision
- Les objectifs de la journée :
  - ▶ Les opérateurs ensemblistes
  - ▶ Les sous-interrogations
  - ▶ La modification des données

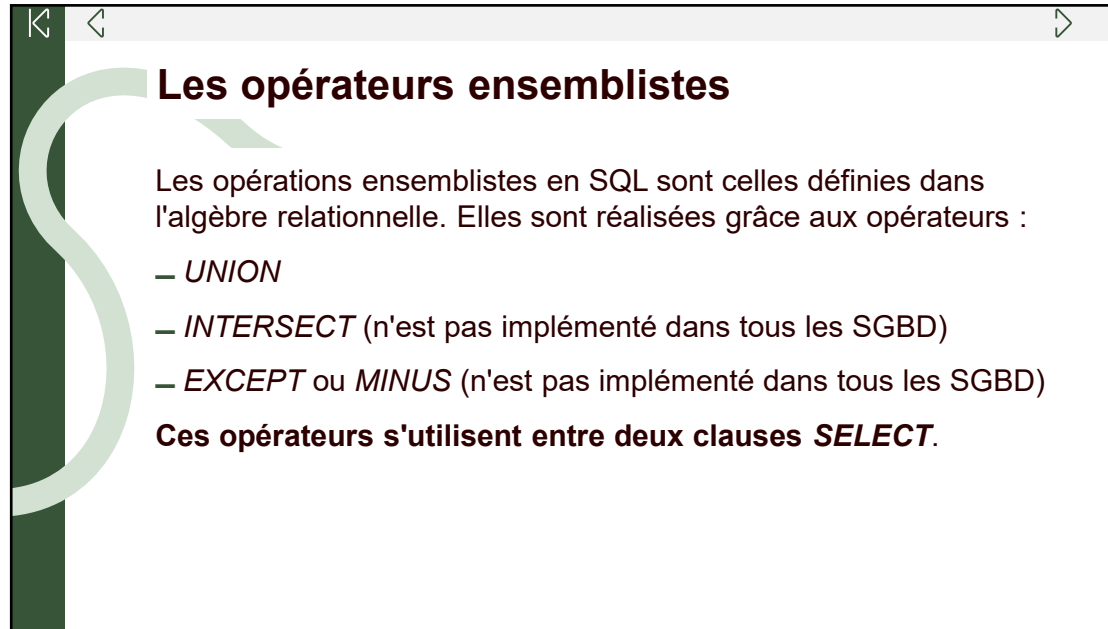


## Plan de la formation

| Module 1 - Introduction           | Module 2 – Le LID                | Module 3 – Fonctions de calcul      | Module 5 – Sous-interrogations  |
|-----------------------------------|----------------------------------|-------------------------------------|---------------------------------|
| Rappels sur le modèle relationnel | La sélection des données         | Les fonctions de calcul             | Les sous-requêtes dans le where |
| Schéma de la BDD du stage         | Atelier                          | Atelier : fonctions de regroupement | Les sous-requêtes dans le from  |
| Caractéristiques du langage SQL   | Les conditions (ou restrictions) | Les clauses Group by et Having      | Atelier                         |
|                                   | Atelier                          | Atelier                             |                                 |
|                                   | Les tris                         |                                     |                                 |
|                                   | Atelier                          | Module 4 – Opérateurs ensemblistes  | Module 6 – Le LMD               |
|                                   |                                  | Les opérateurs ensemblistes         | Modifier les données            |
|                                   |                                  | Atelier                             | Atelier                         |
|                                   |                                  |                                     | Module 7 – Notions sur le LDD   |

Objectifs :

- Exécuter plusieurs requêtes Select en même temps

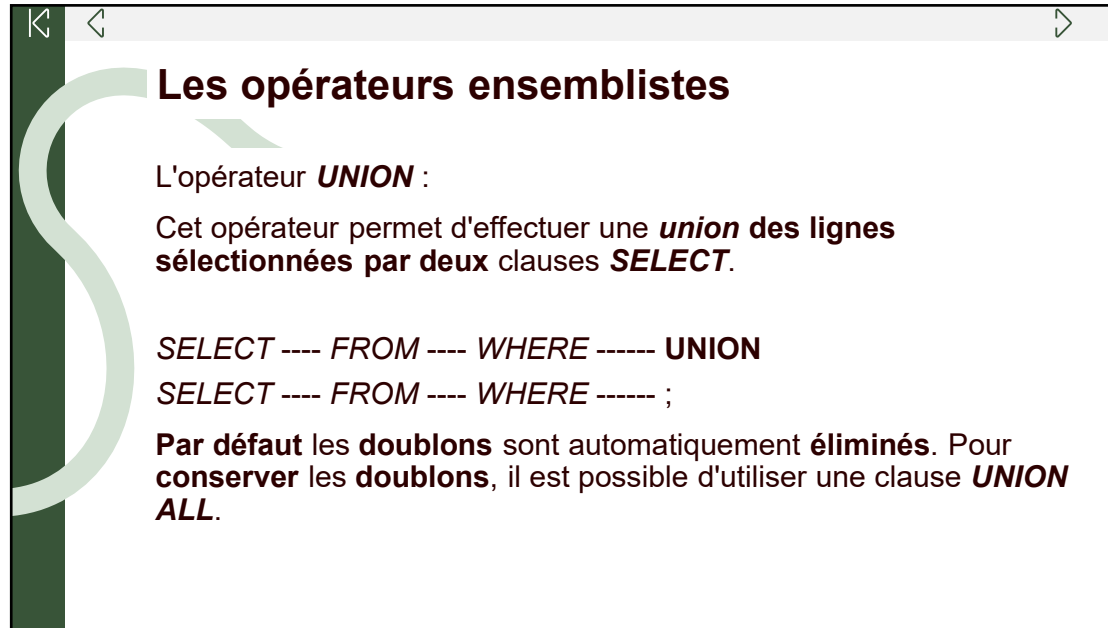


## Les opérateurs ensemblistes

Les opérations ensemblistes en SQL sont celles définies dans l'algèbre relationnelle. Elles sont réalisées grâce aux opérateurs :

- *UNION*
- *INTERSECT* (n'est pas implémenté dans tous les SGBD)
- *EXCEPT* ou *MINUS* (n'est pas implémenté dans tous les SGBD)

**Ces opérateurs s'utilisent entre deux clauses *SELECT*.**



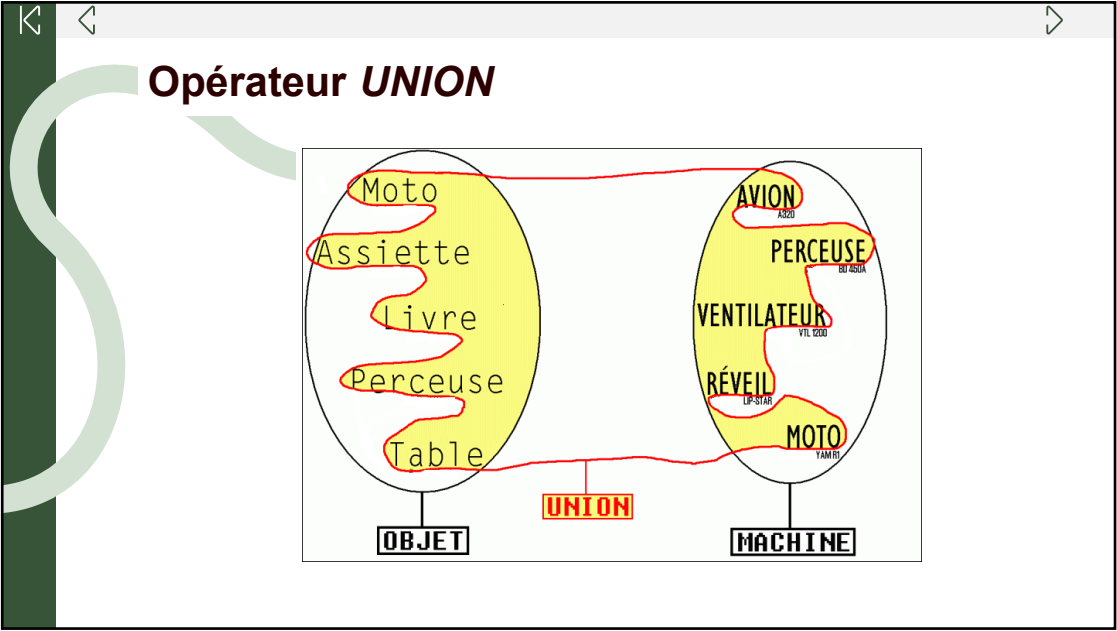
## Les opérateurs ensemblistes

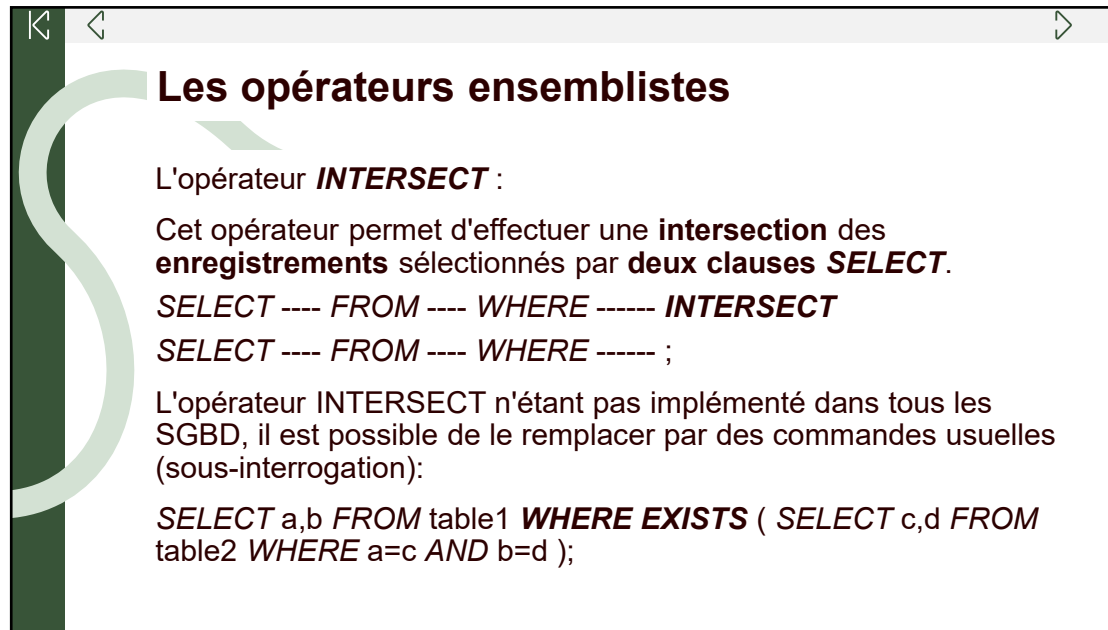
L'opérateur **UNION** :

Cet opérateur permet d'effectuer une **union des lignes sélectionnées par deux clauses *SELECT***.

*SELECT* ---- *FROM* ---- *WHERE* ----- **UNION**  
*SELECT* ---- *FROM* ---- *WHERE* ----- ;

**Par défaut** les **doublons** sont automatiquement **éliminés**. Pour **conserver** les **doublons**, il est possible d'utiliser une clause **UNION ALL**.





## Les opérateurs ensemblistes

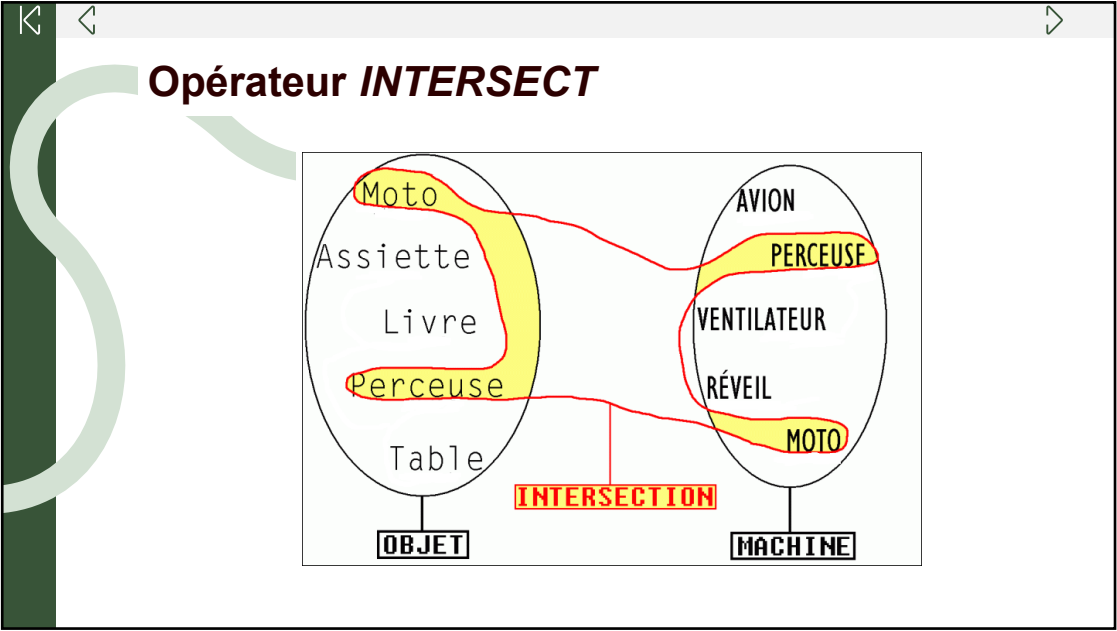
L'opérateur **INTERSECT** :

Cet opérateur permet d'effectuer une **intersection** des **enregistrements** sélectionnés par **deux clauses SELECT**.

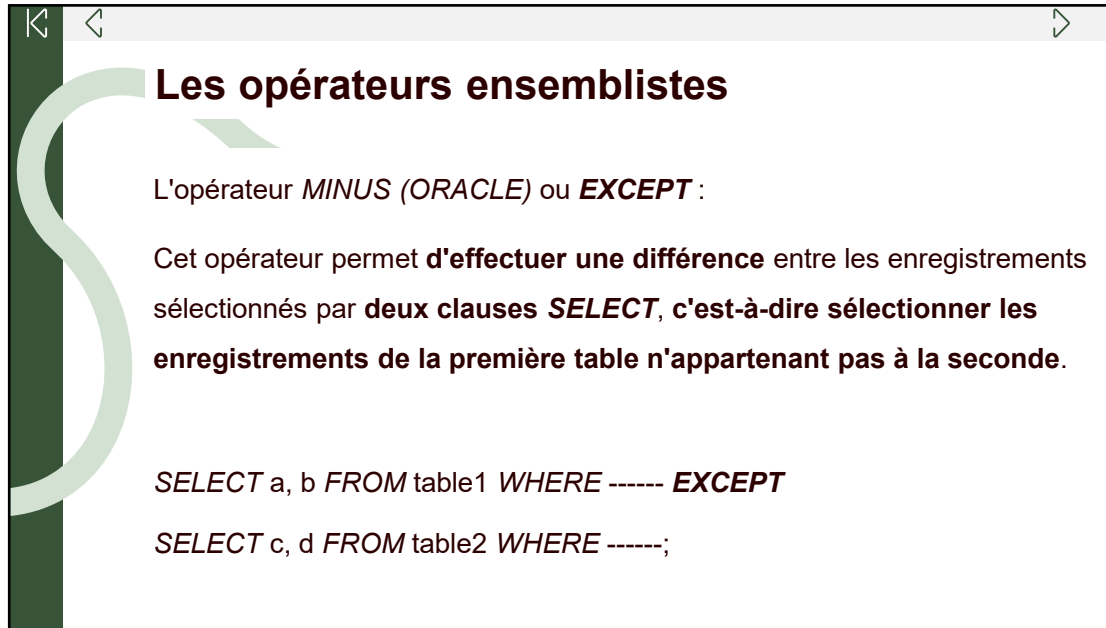
```
SELECT ---- FROM ---- WHERE ----- INTERSECT  
SELECT ---- FROM ---- WHERE ----- ;
```

L'opérateur INTERSECT n'étant pas implémenté dans tous les SGBD, il est possible de le remplacer par des commandes usuelles (sous-interrogation):

```
SELECT a,b FROM table1 WHERE EXISTS ( SELECT c,d FROM  
table2 WHERE a=c AND b=d );
```







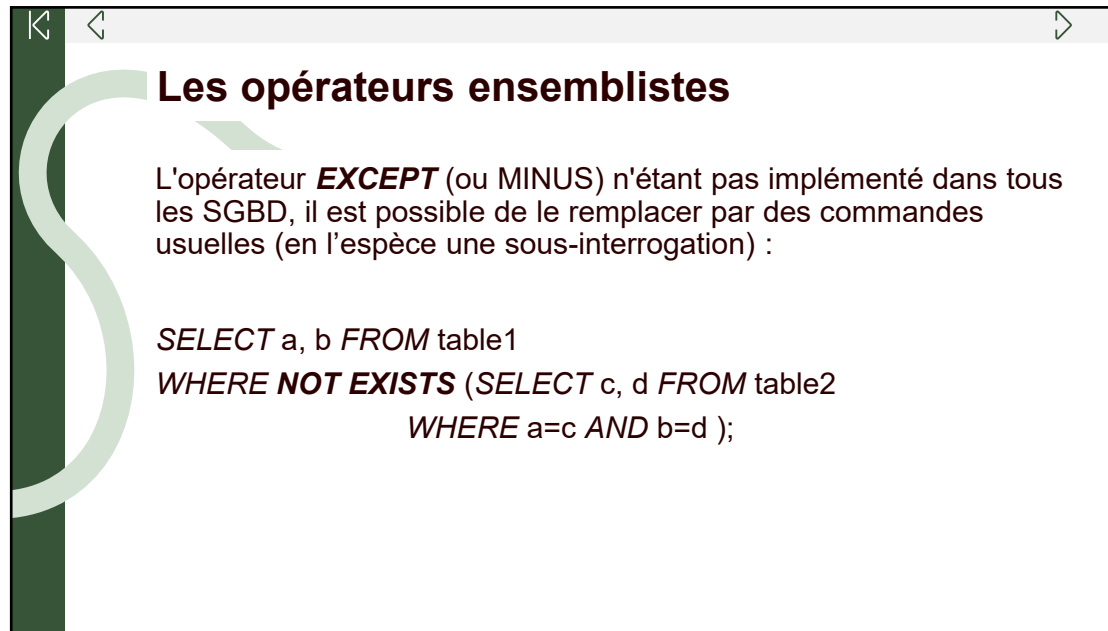
## Les opérateurs ensemblistes

L'opérateur *MINUS* (*ORACLE*) ou ***EXCEPT*** :

Cet opérateur permet **d'effectuer une différence** entre les enregistrements sélectionnés par **deux clauses *SELECT***, c'est-à-dire **sélectionner les enregistrements de la première table n'appartenant pas à la seconde.**

*SELECT* a, b *FROM* table1 *WHERE* ----- ***EXCEPT***

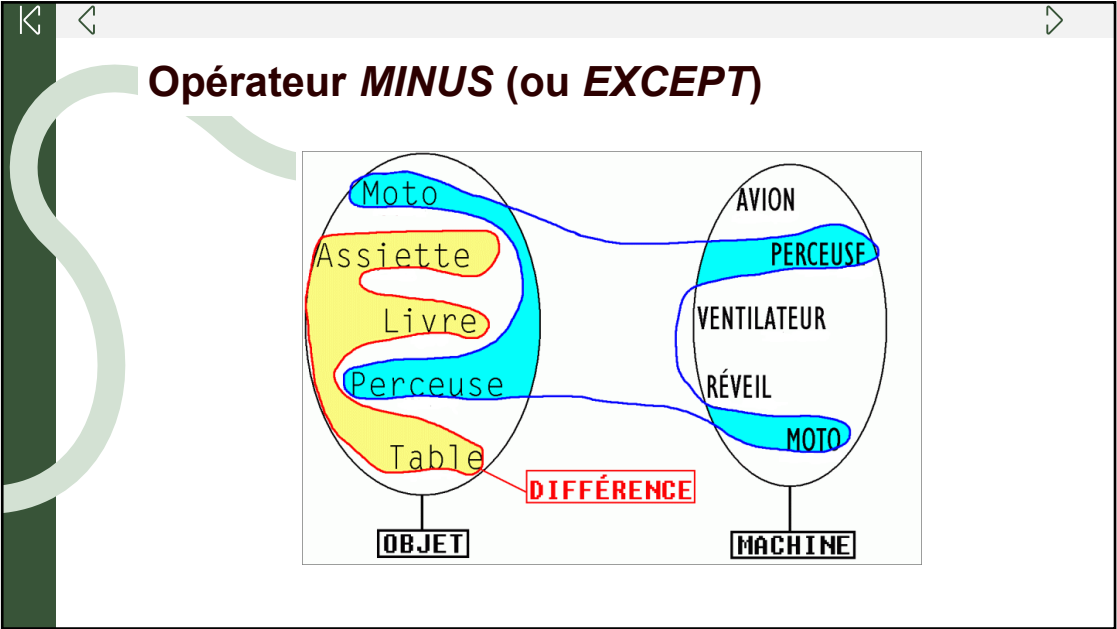
*SELECT* c, d *FROM* table2 *WHERE* -----;



## Les opérateurs ensemblistes

L'opérateur **EXCEPT** (ou MINUS) n'étant pas implémenté dans tous les SGBD, il est possible de le remplacer par des commandes usuelles (en l'espèce une sous-interrogation) :

```
SELECT a, b FROM table1  
WHERE NOT EXISTS (SELECT c, d FROM table2  
                     WHERE a=c AND b=d );
```








**Atelier 4**

Activité

Je reste disponible pour toute question

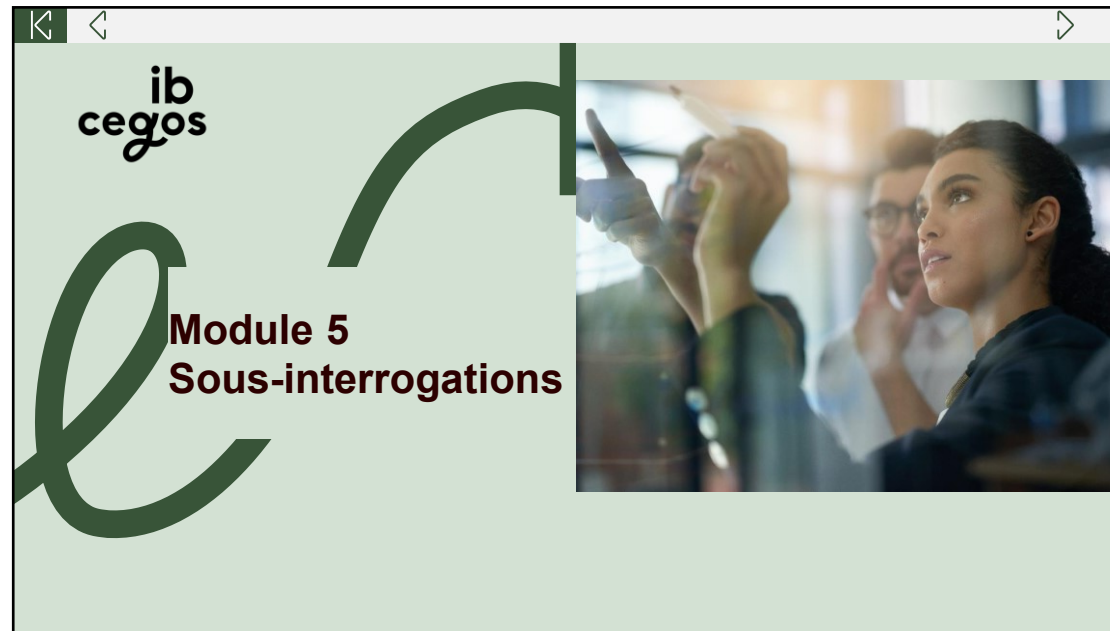
1. Affichez les sociétés, adresse et villes de résidence pour tous les tiers de l'entreprise.  
Trier par société
2. Affichez toutes les commandes qui comportent en même temps des produits de catégorie 1 du fournisseur 1 et des produits de catégorie 2 du fournisseur 2.
3. Affichez la liste des produits que les clients parisiens ne commandent pas.



Ce qu'il faut retenir

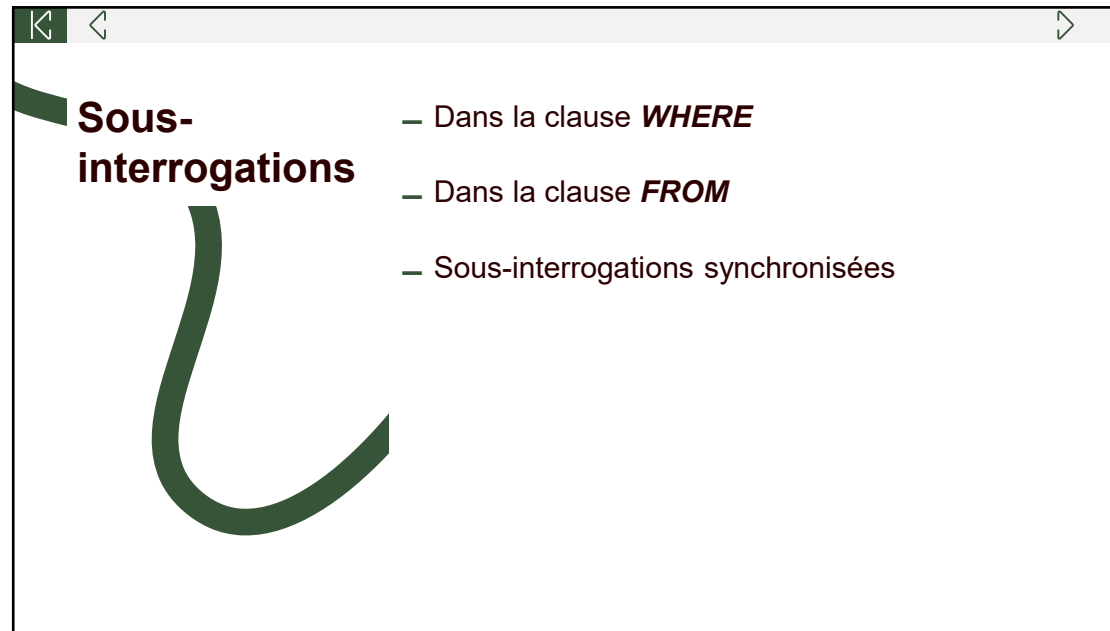
Echanges

- Les *SELECT* des requêtes **assemblées** doivent être synchrones : **même nombre de colonnes, même position.**



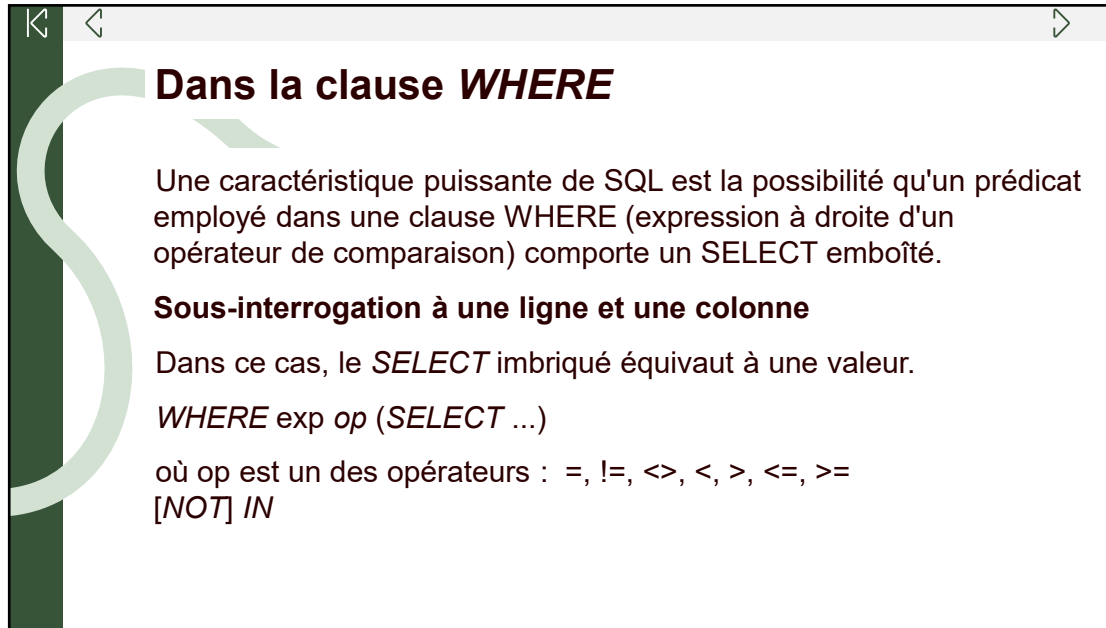
| Plan de la formation              |                                  |  |                                 |
|-----------------------------------|----------------------------------|--|---------------------------------|
| Module 1 - Introduction           | Module 2 – Le LID                | Module 3 – Fonctions de calcul   | Module 5 – Sous-interrogations  |
| Rappels sur le modèle relationnel | La sélection                     | Objectifs : <ul style="list-style-type: none"><li>Intégrer une sous-requête dans la requête principale</li></ul> | Les sous-requêtes dans le where |
| Schéma de la BDD du stage         | Atelier                          |  | Les sous-requêtes dans le from  |
| Caractéristiques du langage SQL   | Les conditions (ou restrictions) | Les clauses Group by et Having   | Atelier                         |
|                                   | Atelier                          | Atelier  |                                 |
|                                   | Les tris                         |  |                                 |
|                                   | Atelier                          | Module 4 – Opérateurs ensemblistes   | Module 6 – Le LMD               |
|                                   | Les jointures entre tables       | Les opérateurs ensemblistes  | Modifier les données            |
|                                   | Atelier                          | Atelier  | Atelier                         |
|                                   |                                  |  | Module 7 – Notions sur le LDD   |





**Sous-interrogations**

- Dans la clause **WHERE**
- Dans la clause **FROM**
- Sous-interrogations synchronisées



## Dans la clause *WHERE*

Une caractéristique puissante de SQL est la possibilité qu'un prédicat employé dans une clause *WHERE* (expression à droite d'un opérateur de comparaison) comporte un *SELECT* emboîté.

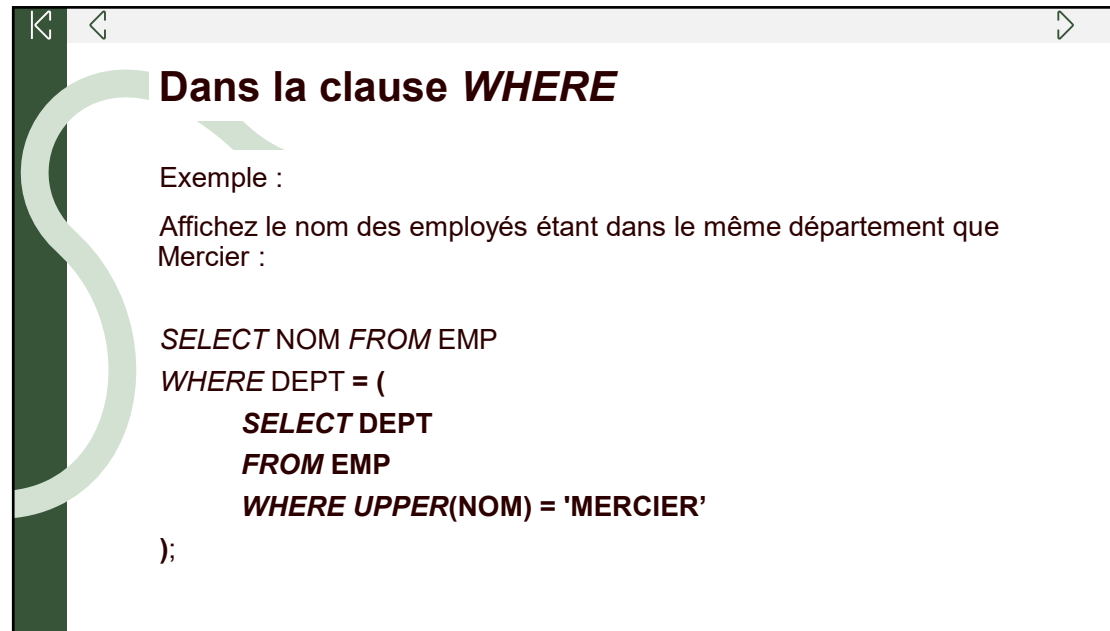
### Sous-interrogation à une ligne et une colonne

Dans ce cas, le *SELECT* imbriqué équivaut à une valeur.

*WHERE* exp op (*SELECT* ...)

où op est un des opérateurs : =, !=, <>, <, >, <=, >=

[*NOT*] *IN*



**Dans la clause *WHERE***

Exemple :

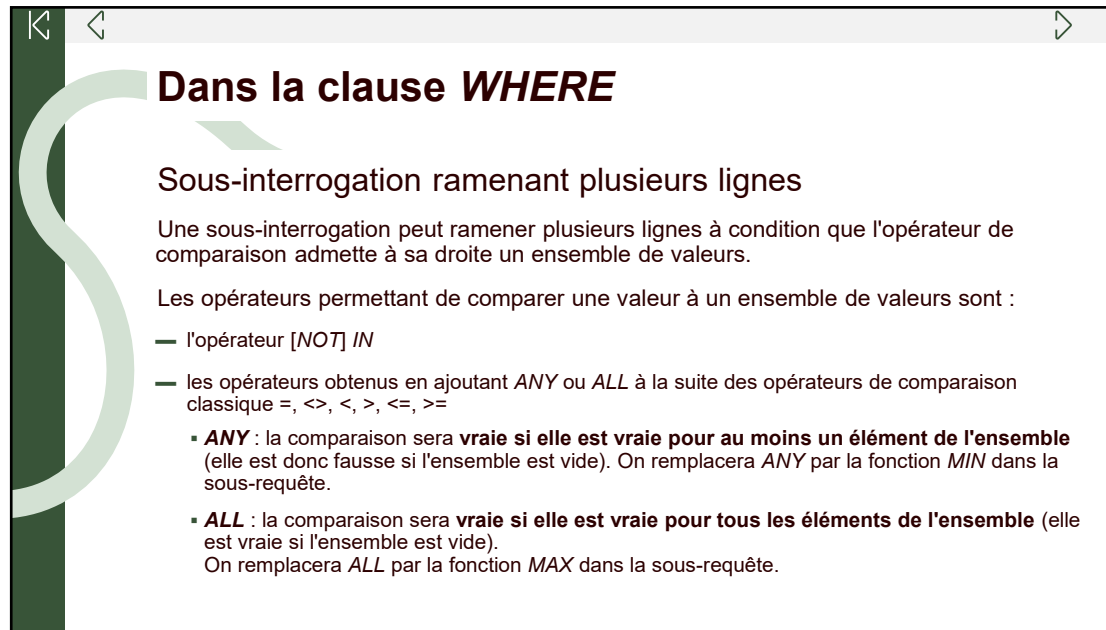
Affichez le nom des employés étant dans le même département que Mercier :

```
SELECT NOM FROM EMP
WHERE DEPT = (
    SELECT DEPT
    FROM EMP
    WHERE UPPER(NOM) = 'MERCIER'
);
```

Exemple :

- Affichez le nom et le salaire des employés ayant un salaire supérieur à la moyenne des salaires

```
SELECT nom, salaire
FROM employees
WHERE salaire > (
    SELECT AVG(salaire) FROM employees
)
```



## Dans la clause *WHERE*

### Sous-interrogation ramenant plusieurs lignes


Une sous-interrogation peut ramener plusieurs lignes à condition que l'opérateur de comparaison admette à sa droite un ensemble de valeurs.

Les opérateurs permettant de comparer une valeur à un ensemble de valeurs sont :

- l'opérateur *[NOT] IN*
- les opérateurs obtenus en ajoutant *ANY* ou *ALL* à la suite des opérateurs de comparaison classique =, <>, <, >, <=, >=
  - **ANY** : la comparaison sera **vraie si elle est vraie pour au moins un élément de l'ensemble** (elle est donc fausse si l'ensemble est vide). On remplacera *ANY* par la fonction *MIN* dans la sous-requête.
  - **ALL** : la comparaison sera **vraie si elle est vraie pour tous les éléments de l'ensemble** (elle est vraie si l'ensemble est vide). On remplacera *ALL* par la fonction *MAX* dans la sous-requête.

- Exemple : Affichez le nom et le prénom des employés ayant un salaire supérieur aux salaires des représentants

```
SELECT nom, prenom FROM employes
WHERE salaire > ALL(
    SELECT salaire FROM employes WHERE fonction LIKE "Repr%"
);
Ou
SELECT nom, prenom FROM employes
WHERE salaire > (
    SELECT MAX(salaire)
    FROM employes
    WHERE fonction LIKE "Repr%"
);
```



# Dans la clause WHERE

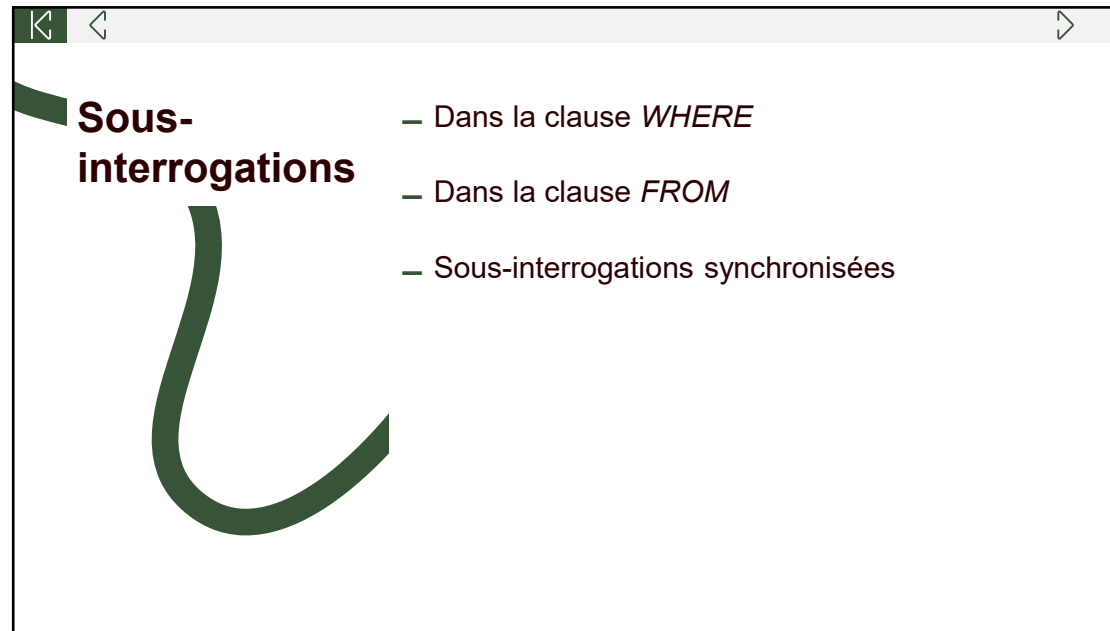
- WHERE exp op ANY (SELECT ...)
- WHERE exp op ALL (SELECT ...)

où op est un des opérateurs =, !=, <>, <, >, <=, >=

- WHERE exp IN (SELECT ...)
- WHERE exp NOT IN (SELECT ...)

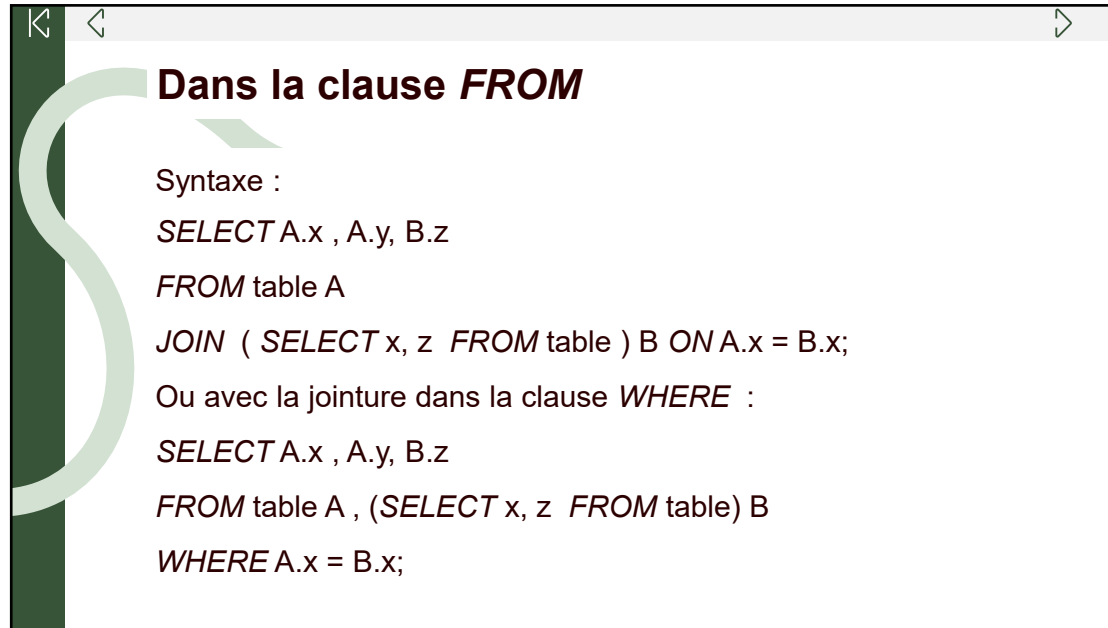
Exemple, liste des employés gagnant plus que tous les employés du département 30 :

```
SELECT NOM, SAL FROM EMP
WHERE SAL > ALL (
  SELECT SAL
  FROM EMP
  WHERE DEPT=30
);
Ou
SELECT NOM, SAL
FROM EMP
WHERE SAL > (
  SELECT MAX(SAL)
  FROM EMP
  WHERE DEPT=30
);
```



**Sous-interrogations**

- Dans la clause *WHERE*
- Dans la clause *FROM*
- Sous-interrogations synchronisées



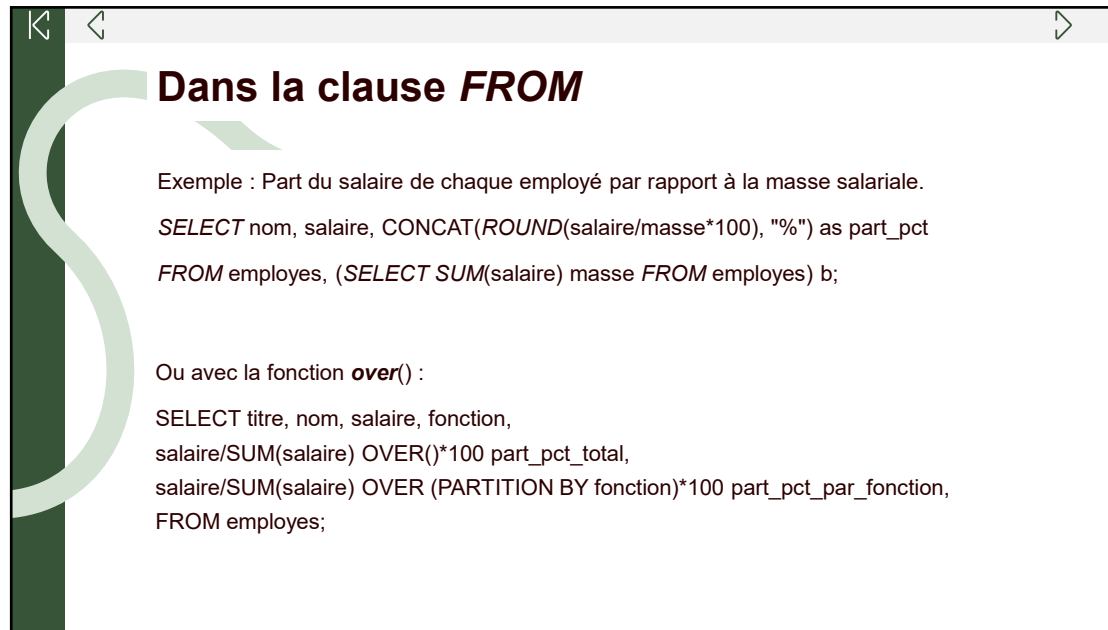
## Dans la clause *FROM*

Syntaxe :

```
SELECT A.x , A.y, B.z  
FROM table A  
JOIN ( SELECT x, z FROM table ) B ON A.x = B.x;
```

Ou avec la jointure dans la clause *WHERE* :

```
SELECT A.x , A.y, B.z  
FROM table A , (SELECT x, z FROM table) B  
WHERE A.x = B.x;
```



## Dans la clause *FROM*

Exemple : Part du salaire de chaque employé par rapport à la masse salariale.

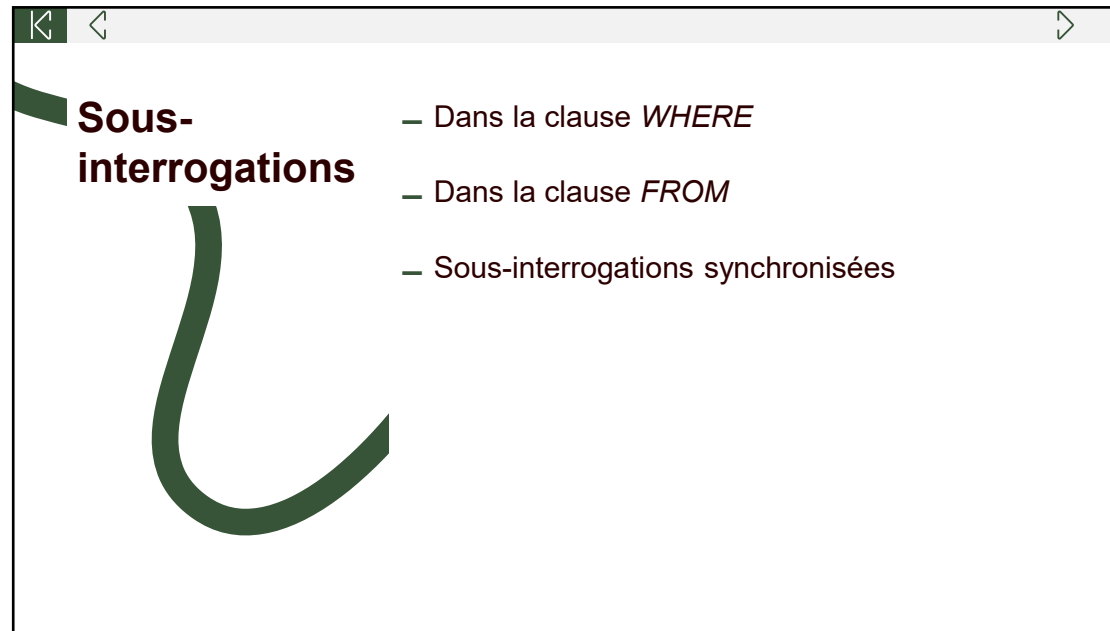
```
SELECT nom, salaire, CONCAT(ROUND(salaire/masse*100), "%") as part_pct  
FROM employes, (SELECT SUM(salaire) masse FROM employes) b;
```

Ou avec la fonction **over()** :

```
SELECT titre, nom, salaire, fonction,  
salaire/SUM(salaire) OVER()*100 part_pct_total,  
salaire/SUM(salaire) OVER (PARTITION BY fonction)*100 part_pct_par_fonction,  
FROM employes;
```

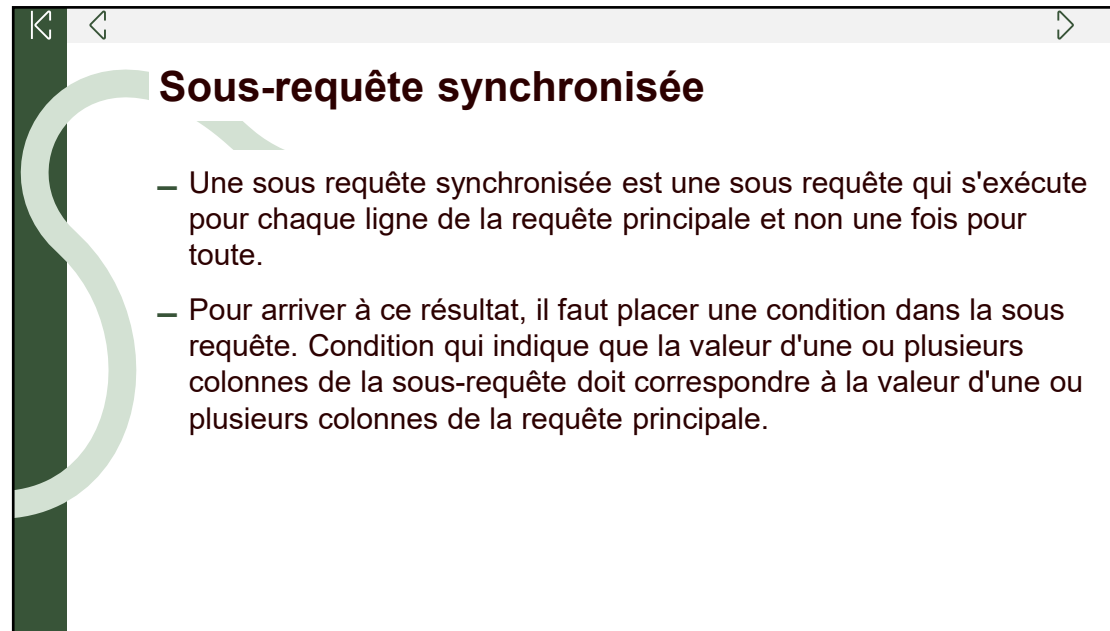
- La fonction OVER() fait partie des fonctions fenêtres de SQL, une fonctionnalité plus avancée permettant d'avoir plus de granularité (précision) par la création des sous-ensembles (partitions) lors de l'utilisation des fonctions d'agrégations.
- Pour avoir plus de détails sur le sujet, vous pouvez consulter cet article <https://learnsql.fr/blog/qu-est-ce-que-la-clause-over-en-sql/>





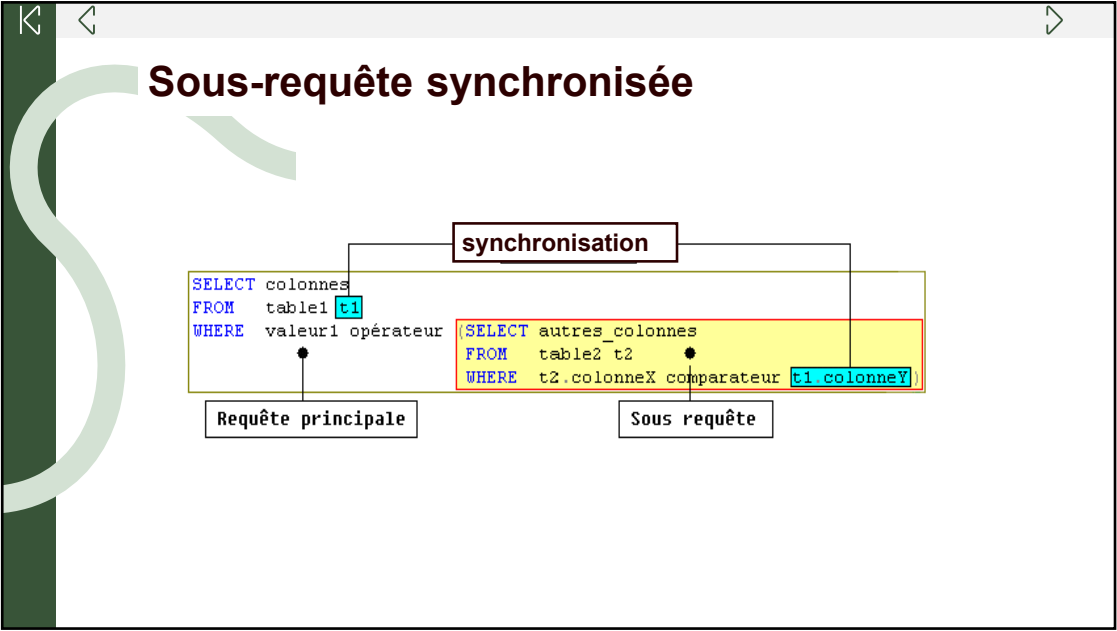
**Sous-interrogations**

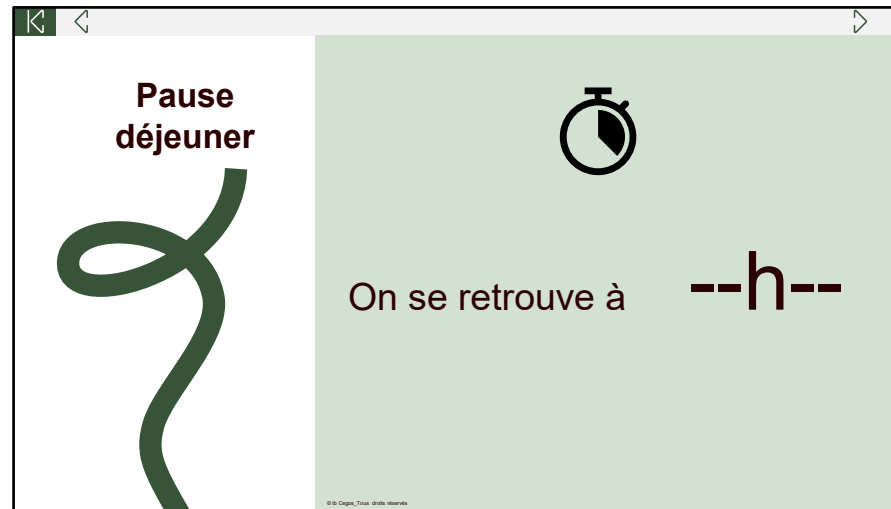
- Dans la clause *WHERE*
- Dans la clause *FROM*
- Sous-interrogations synchronisées

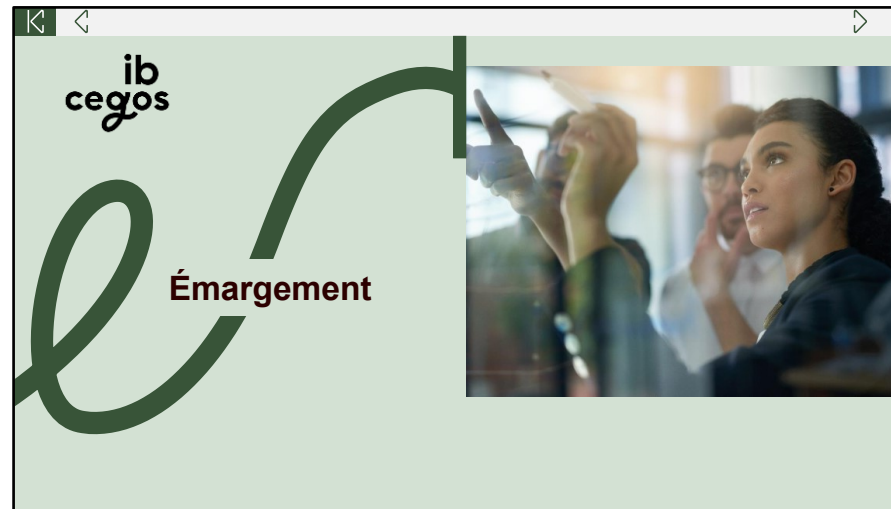


## Sous-requête synchronisée

- Une sous requête synchronisée est une sous requête qui s'exécute pour chaque ligne de la requête principale et non une fois pour toute.
- Pour arriver à ce résultat, il faut placer une condition dans la sous requête. Condition qui indique que la valeur d'une ou plusieurs colonnes de la sous-requête doit correspondre à la valeur d'une ou plusieurs colonnes de la requête principale.







Émargez **chaque début de demi-journée avec une vraie signature** en scannant le **QR Code** projeté dans la salle ou via le lien reçu par votre email d'inscription à cette formation.



Le lien vers l'évaluation est communiqué




**Atelier 5**

Activité

1. Affichez tous les produits pour lesquels la quantité en stock est inférieure à la moyenne des quantités en stock.
2. Affichez toutes les commandes pour lesquelles les frais de ports dépassent la moyenne des frais de ports pour ce client.
3. Affichez les produits pour lesquels la quantité en stock est supérieure à la quantité en stock de tous les produits de catégorie 3.

Je reste disponible pour toute question

Toutes ces requêtes ici illustrent l'utilisation de sous-requête dans le *WHERE*



Ce qu'il faut retenir

Echanges

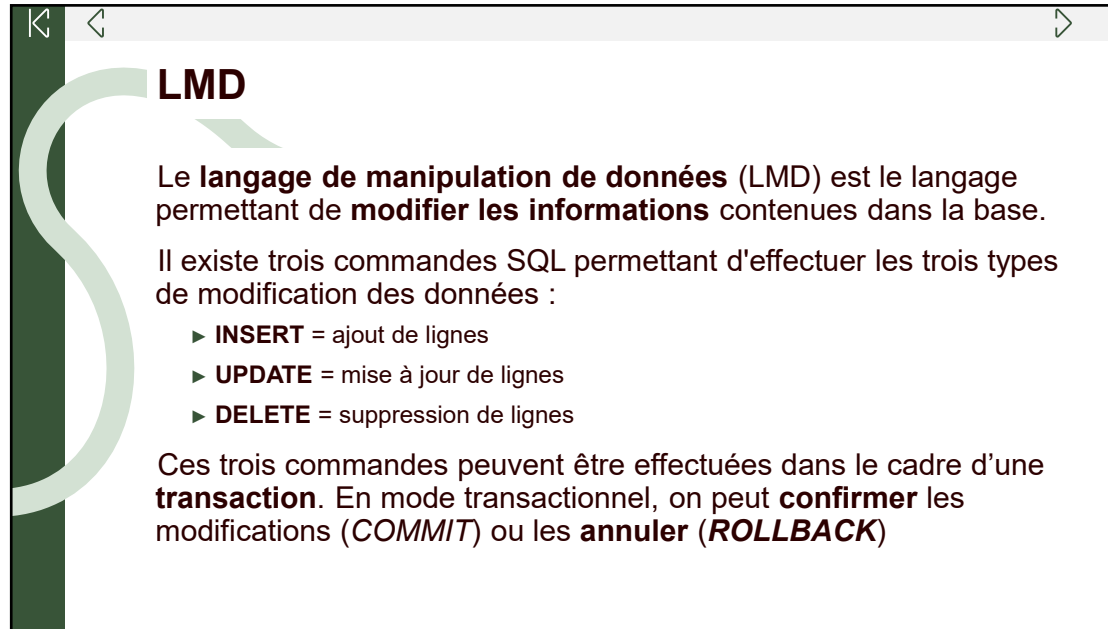
- Une sous-requête dans le **WHERE** permet d'obtenir ou de calculer une information que l'on ne peut pas connaître.
- Une sous-requête dans le **FROM** permet de générer une expression que l'on réutilise ensuite dans la requête principale.





### Plan de la formation

| Module 1 - Introduction           | Module 2 – Le LID  | Module 3 – Fonctions de calcul      | Module 5 – Sous-interrogations  |
|-----------------------------------|--|-------------------------------------|---------------------------------|
| Rappels sur le modèle relationnel | La sélection des données   | Les fonctions de calcul             | Les sous-requêtes dans le where |
| Schéma de la BDD du stage         | Atelier  | Atelier : fonctions de regroupement | Les sous-requêtes dans le from  |
| Caractéristiques du langage SQL   | Les conditions (ou restrictions)   | Les clauses Group by et Having      | Atelier                         |
|                                   | Atelier  | Atelier                             |                                 |
|                                   | Les tris   |                                     |                                 |
|                                   | Atelier  | Module 4 – Opérateurs               | Module 6 – Le LMD               |
| Les jointures                     | Objectifs : <ul style="list-style-type: none"><li>• Insérer, modifier ou supprimer des données</li></ul> |                                     | Modifier les données            |
| Atelier                           |  |                                     | Atelier                         |
|                                   |  |                                     |                                 |
|                                   |  |                                     | Module 7 – Notions sur le LDD   |



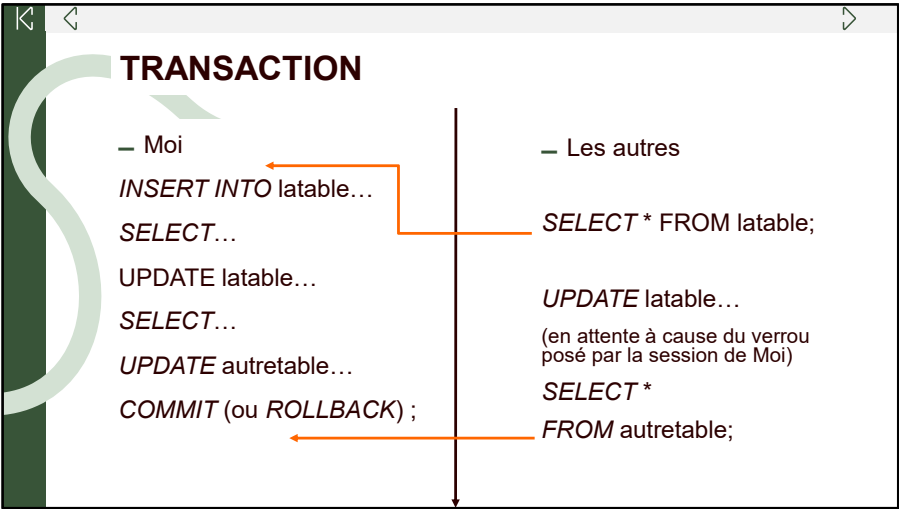
## LMD

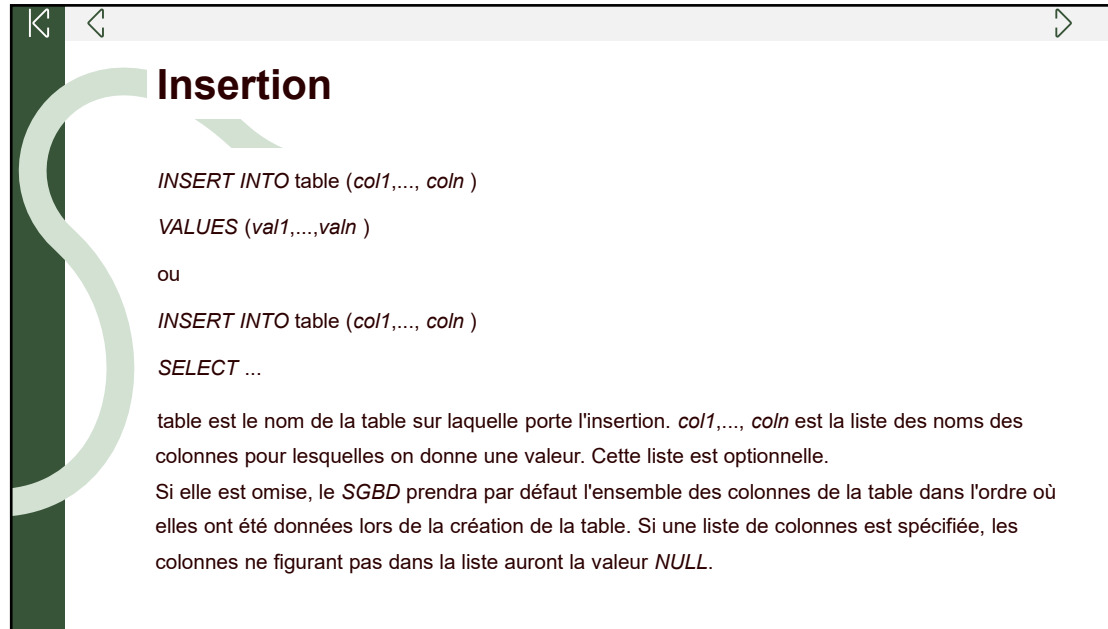
Le **langage de manipulation de données** (LMD) est le langage permettant de **modifier les informations** contenues dans la base.

Il existe trois commandes SQL permettant d'effectuer les trois types de modification des données :

- ▶ **INSERT** = ajout de lignes
- ▶ **UPDATE** = mise à jour de lignes
- ▶ **DELETE** = suppression de lignes

Ces trois commandes peuvent être effectuées dans le cadre d'une **transaction**. En mode transactionnel, on peut **confirmer** les modifications (*COMMIT*) ou les **annuler** (*ROLLBACK*)





## Insertion

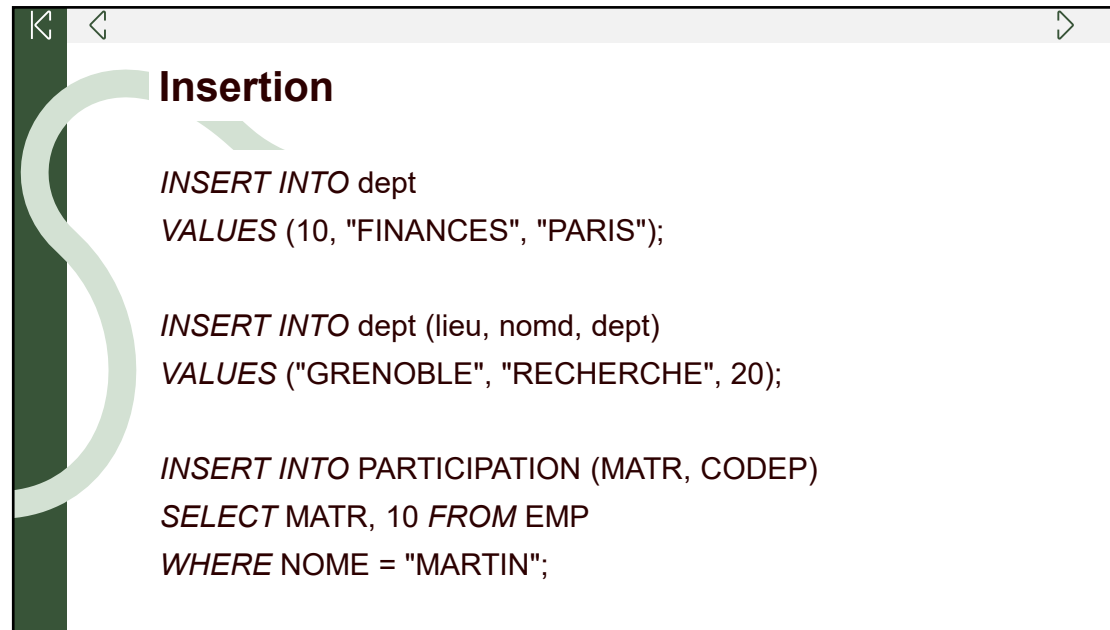
```
INSERT INTO table (col1,..., coln )  
VALUES (val1,...,valn )
```

ou

```
INSERT INTO table (col1,..., coln )  
SELECT ...
```

table est le nom de la table sur laquelle porte l'insertion. *col1,..., coln* est la liste des noms des colonnes pour lesquelles on donne une valeur. Cette liste est optionnelle.

Si elle est omise, le *SGBD* prendra par défaut l'ensemble des colonnes de la table dans l'ordre où elles ont été données lors de la création de la table. Si une liste de colonnes est spécifiée, les colonnes ne figurant pas dans la liste auront la valeur *NULL*.

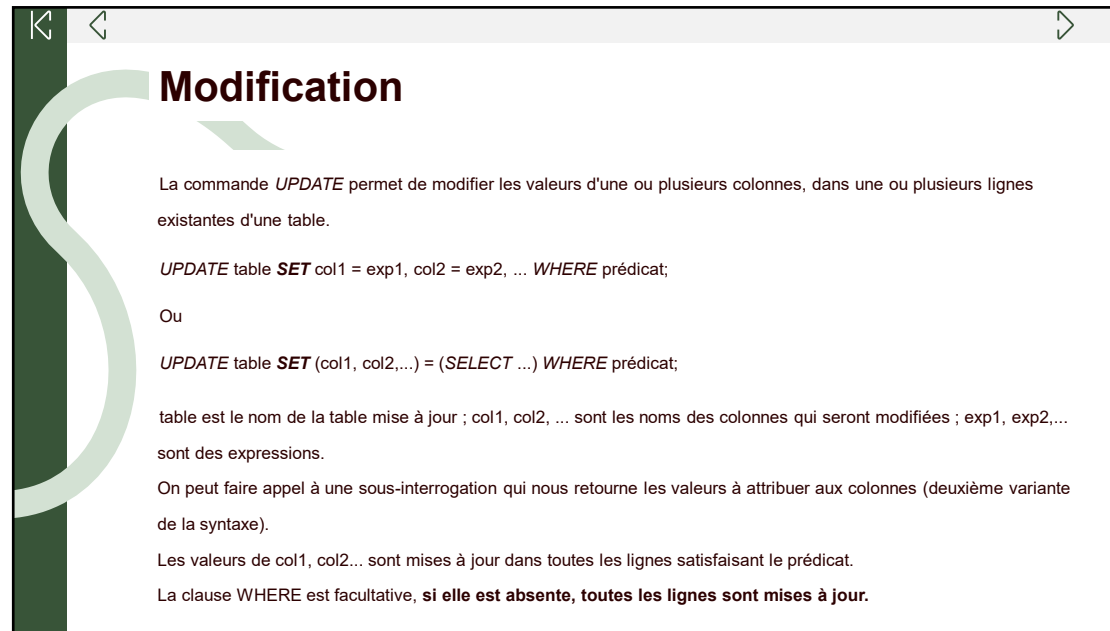


## Insertion

```
INSERT INTO dept  
VALUES (10, "FINANCES", "PARIS");
```

```
INSERT INTO dept (lieu, nomd, dept)  
VALUES ("GRENOBLE", "RECHERCHE", 20);
```

```
INSERT INTO PARTICIPATION (MATR, CODEP)  
SELECT MATR, 10 FROM EMP  
WHERE NOME = "MARTIN";
```



## Modification

La commande *UPDATE* permet de modifier les valeurs d'une ou plusieurs colonnes, dans une ou plusieurs lignes existantes d'une table.

*UPDATE* table **SET** col1 = exp1, col2 = exp2, ... *WHERE* prédicat;

Ou

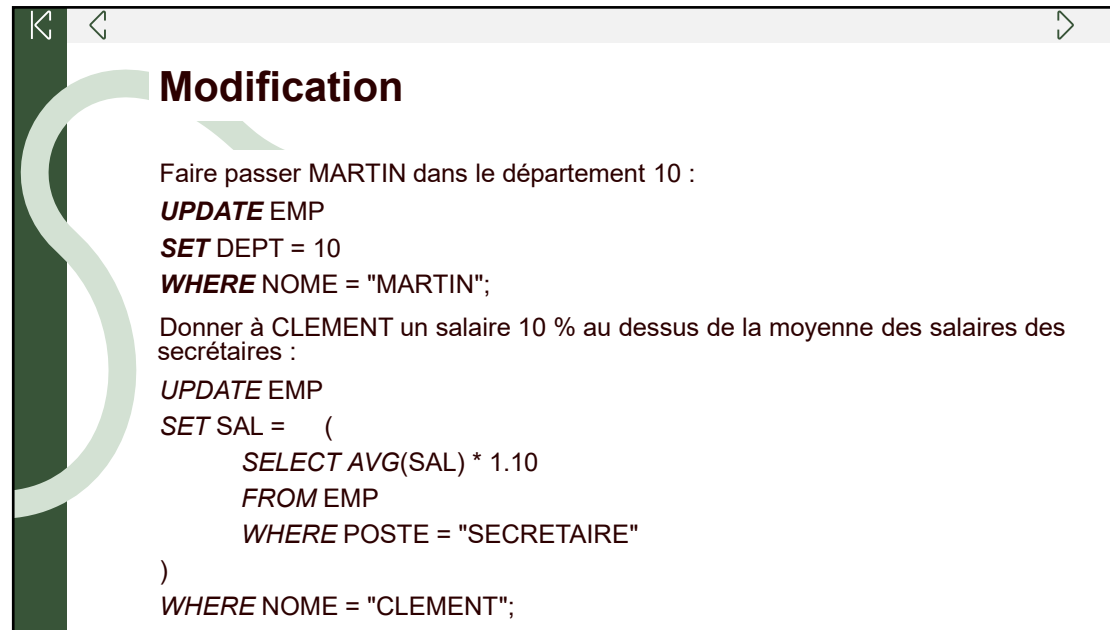
*UPDATE* table **SET** (col1, col2,...) = (*SELECT* ...) *WHERE* prédicat;

table est le nom de la table mise à jour ; col1, col2, ... sont les noms des colonnes qui seront modifiées ; exp1, exp2,... sont des expressions.

On peut faire appel à une sous-interrogation qui nous retourne les valeurs à attribuer aux colonnes (deuxième variante de la syntaxe).

Les valeurs de col1, col2... sont mises à jour dans toutes les lignes satisfaisant le prédicat.

La clause *WHERE* est facultative, **si elle est absente, toutes les lignes sont mises à jour.**



## Modification

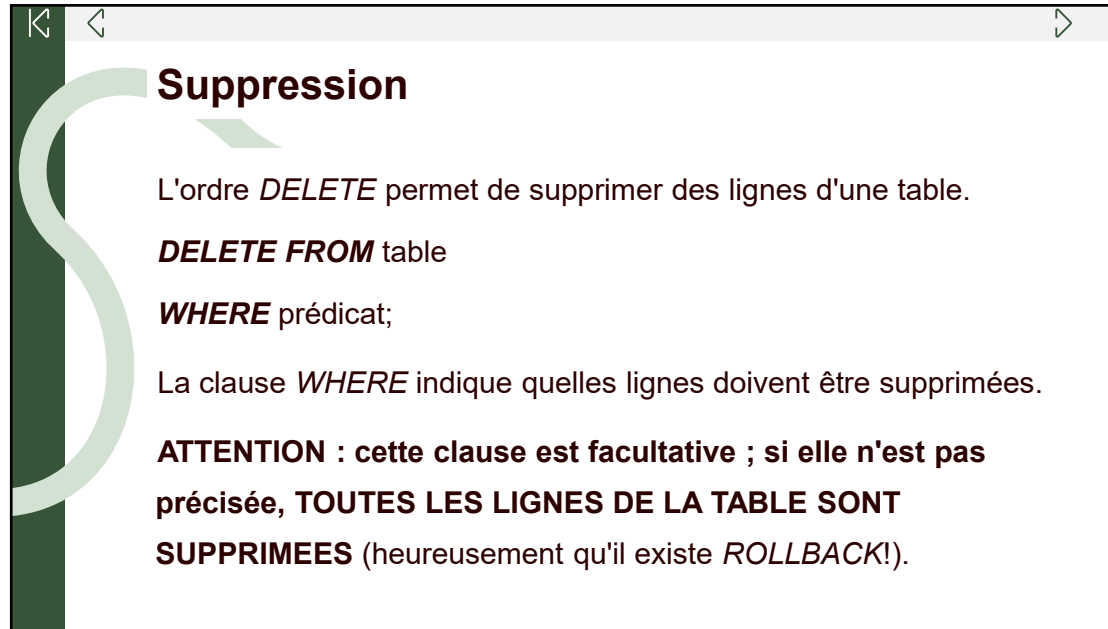
Faire passer MARTIN dans le département 10 :

```
UPDATE EMP
SET DEPT = 10
WHERE NOME = "MARTIN";
```

Donner à CLEMENT un salaire 10 % au dessus de la moyenne des salaires des secrétaires :

```
UPDATE EMP
SET SAL = (
    SELECT AVG(SAL) * 1.10
    FROM EMP
    WHERE POSTE = "SECRETAIRE"
)
WHERE NOME = "CLEMENT";
```





## Suppression

L'ordre *DELETE* permet de supprimer des lignes d'une table.

***DELETE FROM*** table

***WHERE*** prédicat;

La clause *WHERE* indique quelles lignes doivent être supprimées.

**ATTENTION : cette clause est facultative ; si elle n'est pas précisée, TOUTES LES LIGNES DE LA TABLE SONT SUPPRIMEES** (heureusement qu'il existe *ROLLBACK!*).



## Suppression

```
DELETE FROM dept  
WHERE dept = 10;
```

*DELETE FROM dept; <= j'efface toutes les lignes*

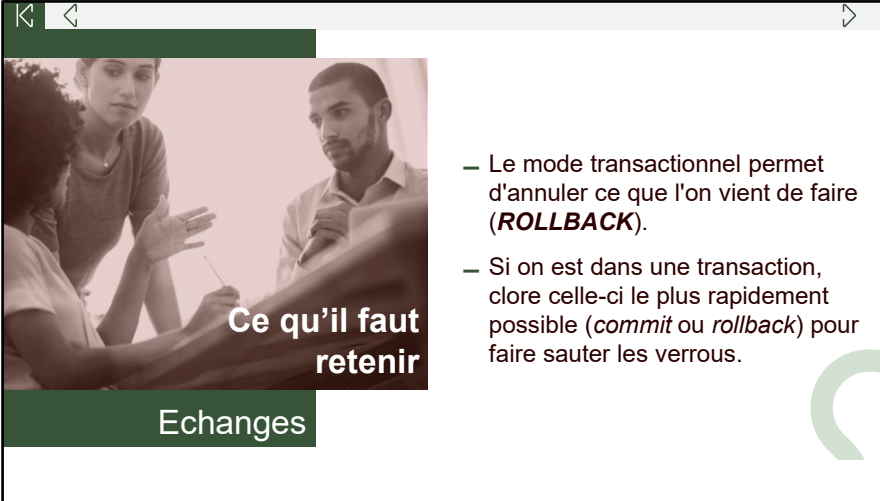


**Atelier 6**

Activité

Je reste disponible pour toute question

1. Insérez une nouvelle catégorie de produits nommée «fruits et légumes », en respectant les contraintes.
2. Créez un nouveau fournisseur « Grandma » (no\_fournisseur = 30) avec les mêmes coordonnées que le fournisseur « Grandma Kelly's Homestead ».
3. Attribuer les produits de « Grandma Kelly's Homestead » au nouveau fournisseur précédemment créé.
4. Supprimez l'ancien fournisseur « Grandma Kelly's Homestead » .

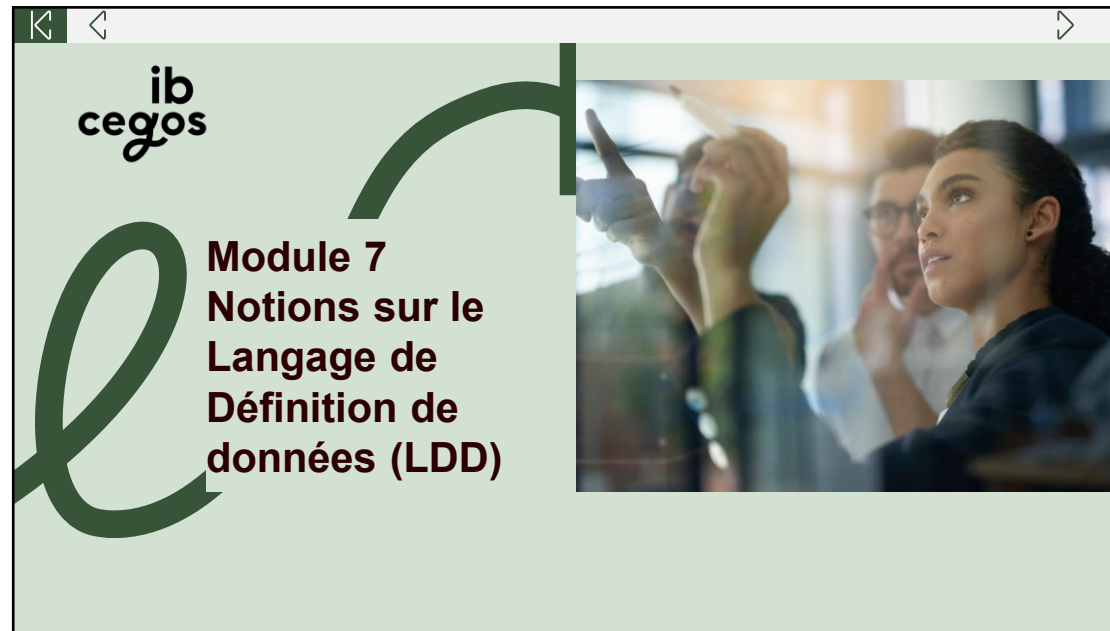


**Ce qu'il faut retenir**

Echanges

- Le mode transactionnel permet d'annuler ce que l'on vient de faire (**ROLLBACK**).
- Si on est dans une transaction, close celle-ci le plus rapidement possible (*commit* ou *rollback*) pour faire sauter les verrous.





The slide features a light green background. In the top left corner, there is a navigation bar with a back arrow, a forward arrow, and a search icon. The ib cegos logo is positioned in the upper left. A large, stylized green letter 'e' is on the left side. The main title, 'Module 7 Notions sur le Langage de Définition de données (LDD)', is centered in a dark brown font. On the right, there is a photograph of three people in a professional setting; a woman in the foreground is pointing upwards, while two men behind her are also looking in the same direction.

**ib  
cegos**

**Module 7**  
**Notions sur le**  
**Langage de**  
**Définition de**  
**données (LDD)**

| Plan de la formation              |                                  |                                     |                                 |
|-----------------------------------|----------------------------------|-------------------------------------|---------------------------------|
| Module 1 - Introduction           | Module 2 – Le LID                | Module 3 – Fonctions de calcul      | Module 5 – Sous-interrogations  |
| Rappels sur le modèle relationnel | La sélection des données         | Les fonctions de calcul             | Les sous-requêtes dans le where |
| Schéma de la BDD du stage         | Atelier                          | Atelier : fonctions de regroupement | Les sous-requêtes dans le from  |
| Caractéristiques du langage SQL   | Les conditions (ou restrictions) | Les clauses Group by et Having      | Atelier                         |
|                                   | Atelier                          | Atelier                             |                                 |
|                                   | Les tris                         |                                     |                                 |
|                                   | Atelier                          | Module 4 – Opérateurs ensemblistes  | Module 6 – Le LMD               |
|                                   | Les jointures entre tables       | Les opérateurs ensemblistes         | Modifier les données            |
|                                   | Atelier                          | Atelier                             | Atelier                         |
|                                   |                                  |                                     | Module 7 – Notions sur le LDD   |

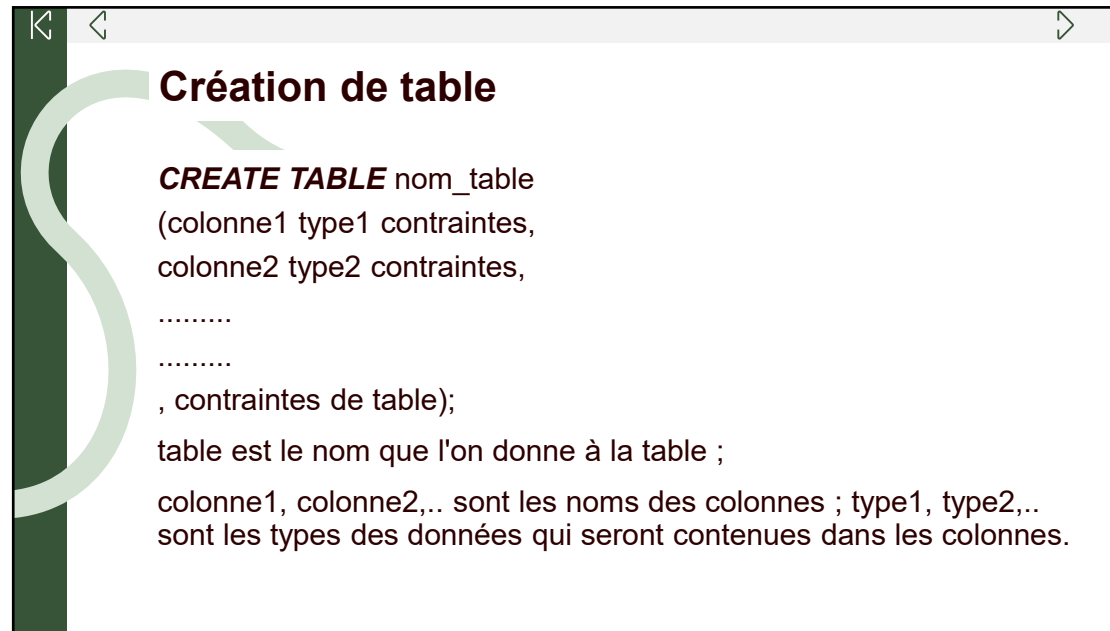
Objectifs :

- Table, vue : quelle est la différence ?
- A quoi sert un index ?

# Types de données

| TYPE TEXTE     |
|----------------|
| CHAR           |
| VARCHAR        |
| LONG           |
| ...            |
| TYPE NUMERIQUE |
| NUMBER         |
| INTEGER        |
| FLOAT          |
| BLOB, BIT      |
| INTEGER        |
| ...            |
| TYPE DATE      |
| DATE           |
| ...            |



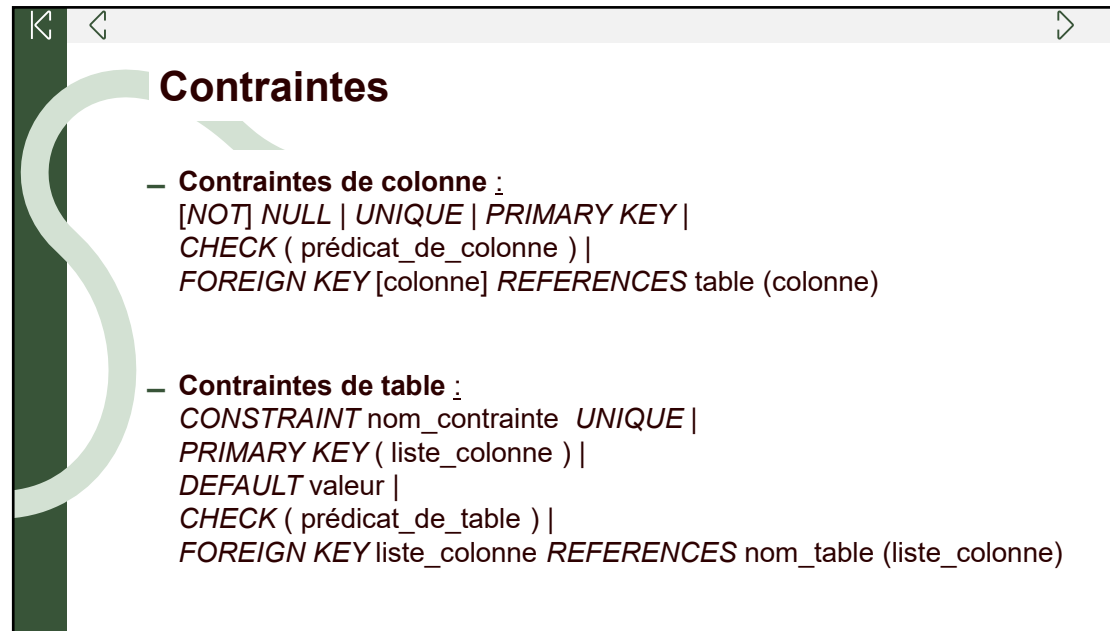


## Création de table

```
CREATE TABLE nom_table  
(colonne1 type1 contraintes,  
colonne2 type2 contraintes,  
.....  
.....  
, contraintes de table);
```

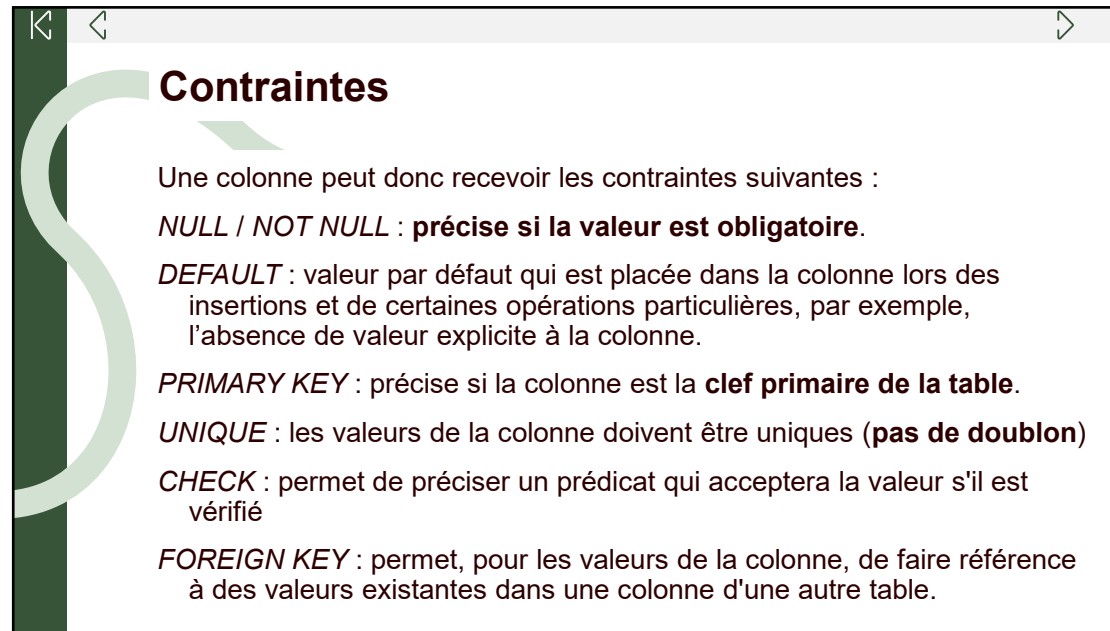
table est le nom que l'on donne à la table ;

colonne1, colonne2,.. sont les noms des colonnes ; type1, type2,..  
sont les types des données qui seront contenues dans les colonnes.



## Contraintes

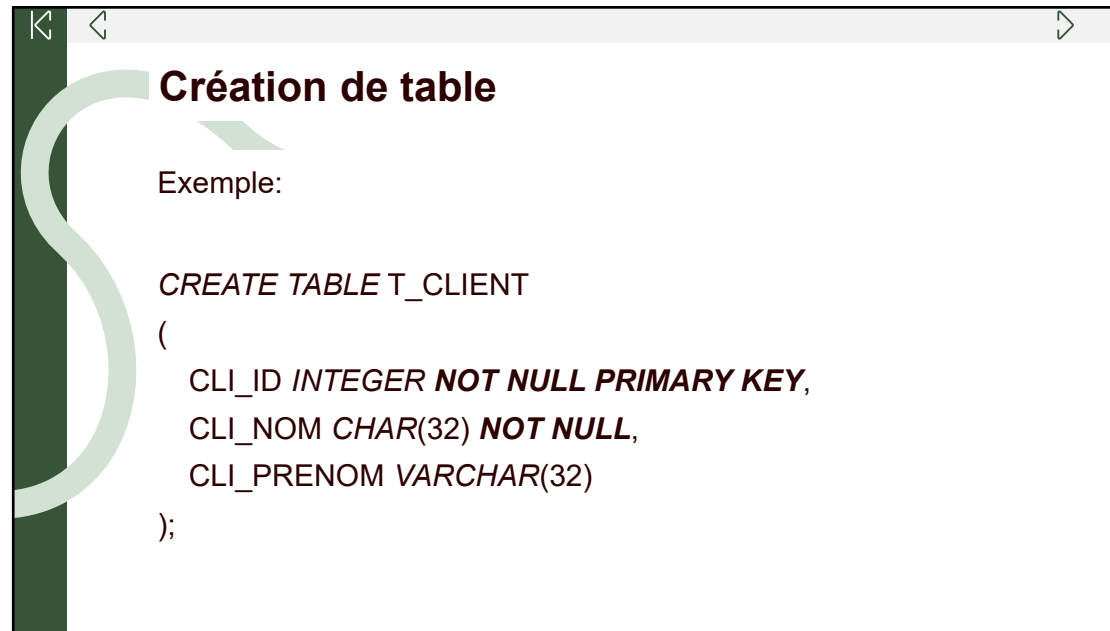
- **Contraintes de colonne :**  
`[NOT] NULL` | `UNIQUE` | `PRIMARY KEY` |  
`CHECK ( prédicat_de_colonne )` |  
`FOREIGN KEY [colonne] REFERENCES table (colonne)`
- **Contraintes de table :**  
`CONSTRAINT nom_contrainte UNIQUE` |  
`PRIMARY KEY ( liste_colonne )` |  
`DEFAULT valeur` |  
`CHECK ( prédicat_de_table )` |  
`FOREIGN KEY liste_colonne REFERENCES nom_table (liste_colonne)`



## Contraintes

Une colonne peut donc recevoir les contraintes suivantes :

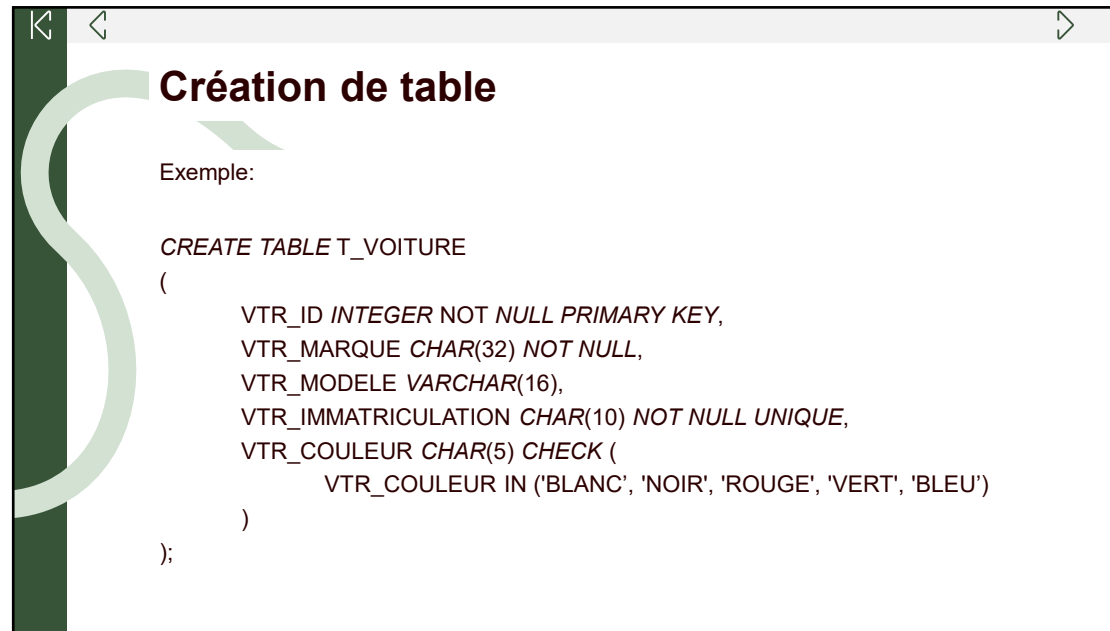
- NULL / NOT NULL*** : précise si la valeur est obligatoire.
- DEFAULT*** : valeur par défaut qui est placée dans la colonne lors des insertions et de certaines opérations particulières, par exemple, l'absence de valeur explicite à la colonne.
- PRIMARY KEY*** : précise si la colonne est la **clef primaire de la table**.
- UNIQUE*** : les valeurs de la colonne doivent être uniques (**pas de doublon**)
- CHECK*** : permet de préciser un prédicat qui acceptera la valeur s'il est vérifié
- FOREIGN KEY*** : permet, pour les valeurs de la colonne, de faire référence à des valeurs existantes dans une colonne d'une autre table.



## Création de table

Exemple:

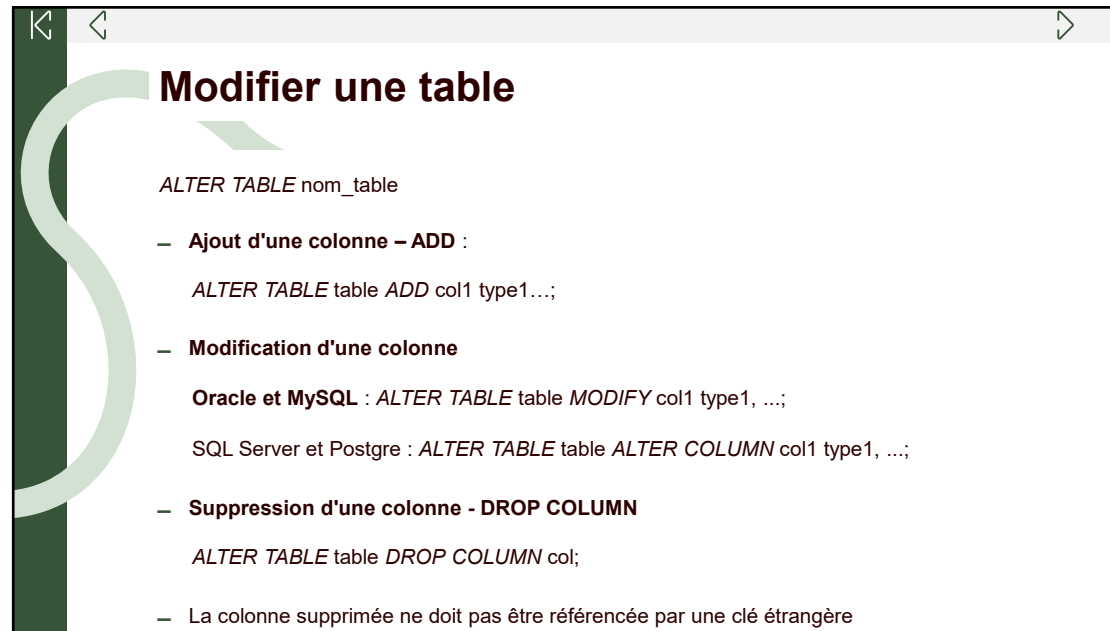
```
CREATE TABLE T_CLIENT  
(  
    CLI_ID INTEGER NOT NULL PRIMARY KEY,  
    CLI_NOM CHAR(32) NOT NULL,  
    CLI_PRENOM VARCHAR(32)  
);
```



## Création de table

Exemple:

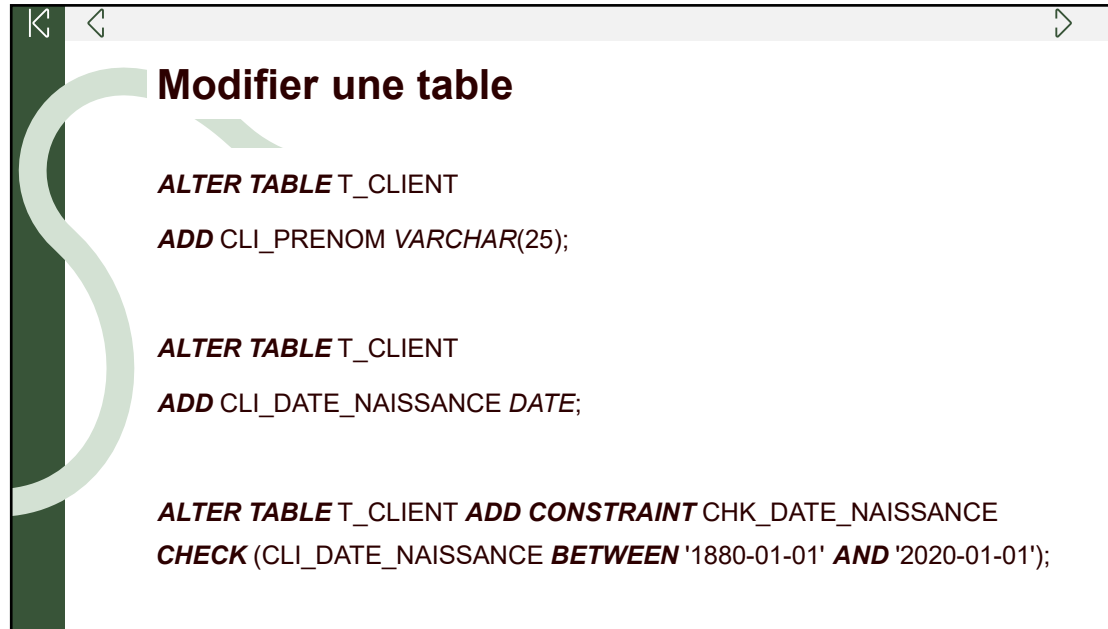
```
CREATE TABLE T_VOITURE
(
    VTR_ID INTEGER NOT NULL PRIMARY KEY,
    VTR_MARQUE CHAR(32) NOT NULL,
    VTR_MODELE VARCHAR(16),
    VTR_IMMATRICULATION CHAR(10) NOT NULL UNIQUE,
    VTR_COULEUR CHAR(5) CHECK (
        VTR_COULEUR IN ('BLANC', 'NOIR', 'ROUGE', 'VERT', 'BLEU')
    )
);
```



## Modifier une table

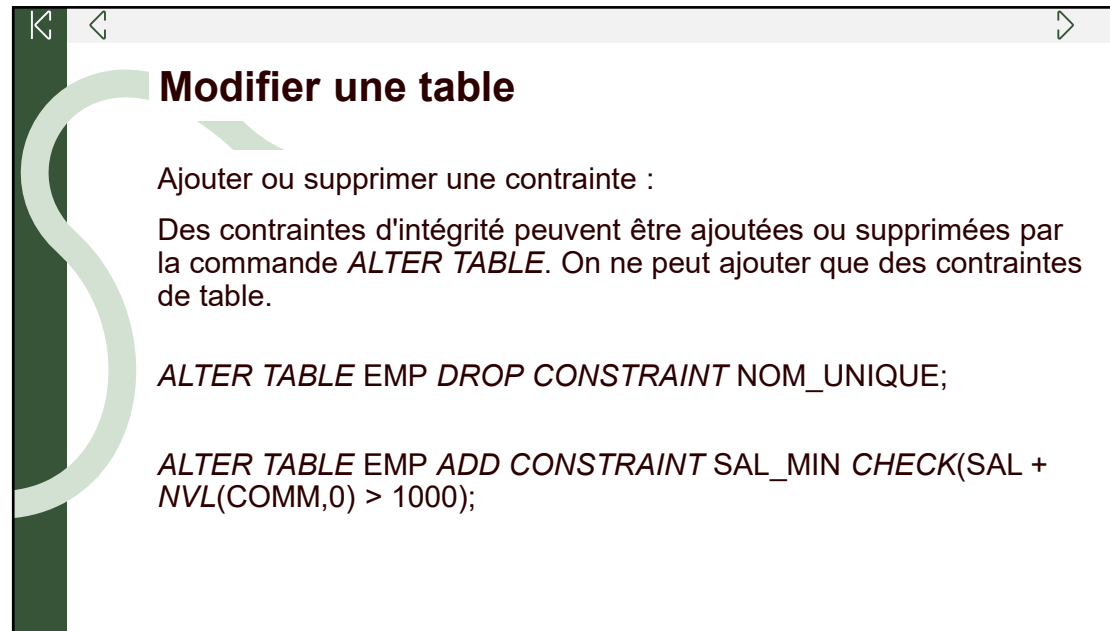
*ALTER TABLE* nom\_table

- **Ajout d'une colonne – ADD :**  
*ALTER TABLE* table *ADD* col1 type1...;
- **Modification d'une colonne**  
**Oracle et MySQL :** *ALTER TABLE* table *MODIFY* col1 type1, ...;  
**SQL Server et Postgre :** *ALTER TABLE* table *ALTER COLUMN* col1 type1, ...;
- **Suppression d'une colonne - DROP COLUMN**  
*ALTER TABLE* table *DROP COLUMN* col;
- La colonne supprimée ne doit pas être référencée par une clé étrangère



**Modifier une table**

```
ALTER TABLE T_CLIENT  
ADD CLI_PRENOM VARCHAR(25);  
  
ALTER TABLE T_CLIENT  
ADD CLI_DATE_NAISSANCE DATE;  
  
ALTER TABLE T_CLIENT ADD CONSTRAINT CHK_DATE_NAISSANCE  
CHECK (CLI_DATE_NAISSANCE BETWEEN '1880-01-01' AND '2020-01-01');
```



## Modifier une table

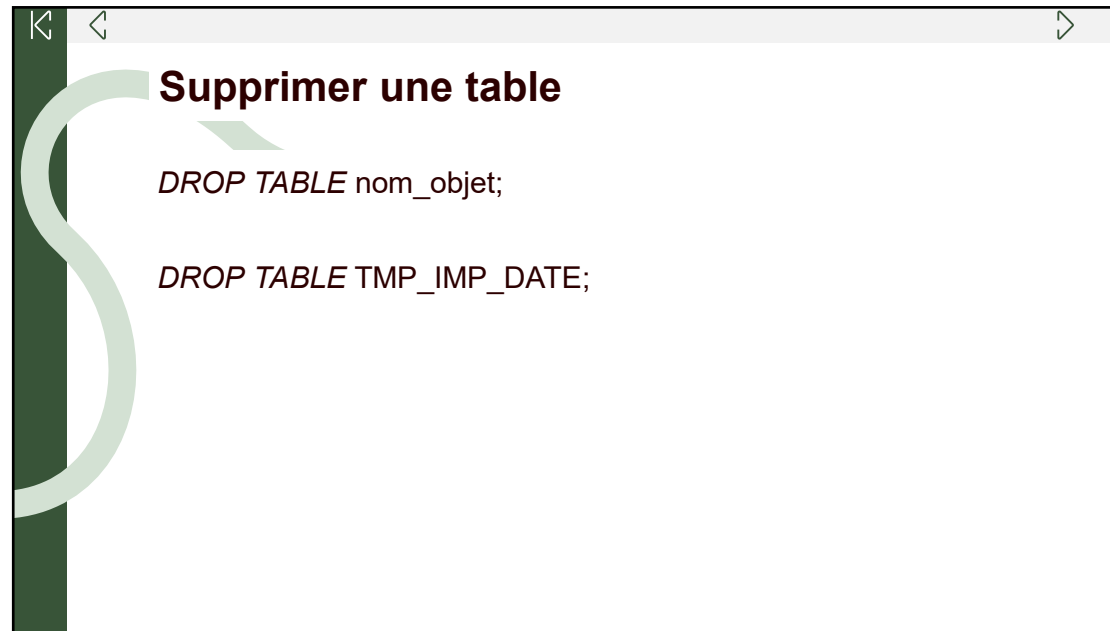
Ajouter ou supprimer une contrainte :

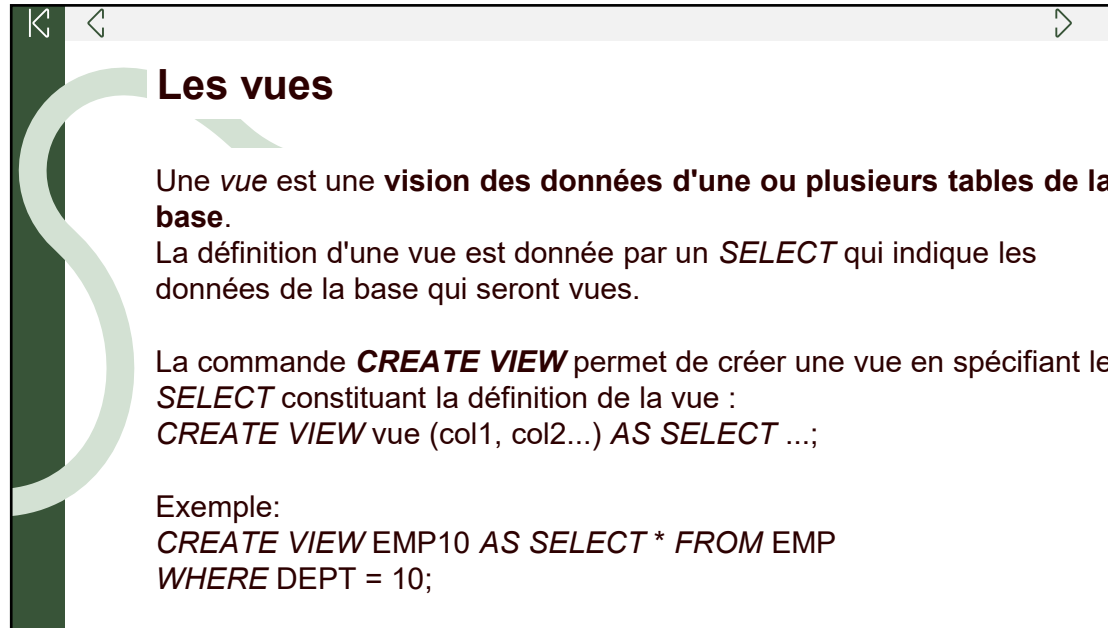
Des contraintes d'intégrité peuvent être ajoutées ou supprimées par la commande *ALTER TABLE*. On ne peut ajouter que des contraintes de table.

```
ALTER TABLE EMP DROP CONSTRAINT NOM_UNIQUE;
```

```
ALTER TABLE EMP ADD CONSTRAINT SAL_MIN CHECK(SAL +  
NVL(COMM,0) > 1000);
```







## Les vues

Une *vue* est une **vision des données d'une ou plusieurs tables de la base**.

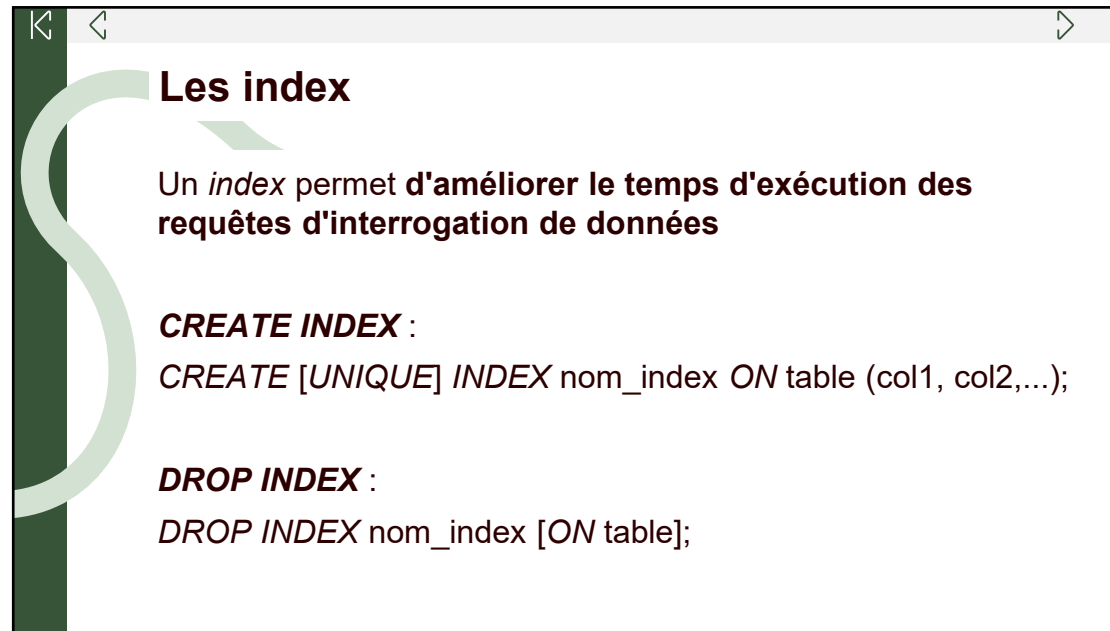
La définition d'une vue est donnée par un *SELECT* qui indique les données de la base qui seront vues.

La commande **CREATE VIEW** permet de créer une vue en spécifiant le *SELECT* constituant la définition de la vue :

```
CREATE VIEW vue (col1, col2...) AS SELECT ...;
```

Exemple:

```
CREATE VIEW EMP10 AS SELECT * FROM EMP  
WHERE DEPT = 10;
```



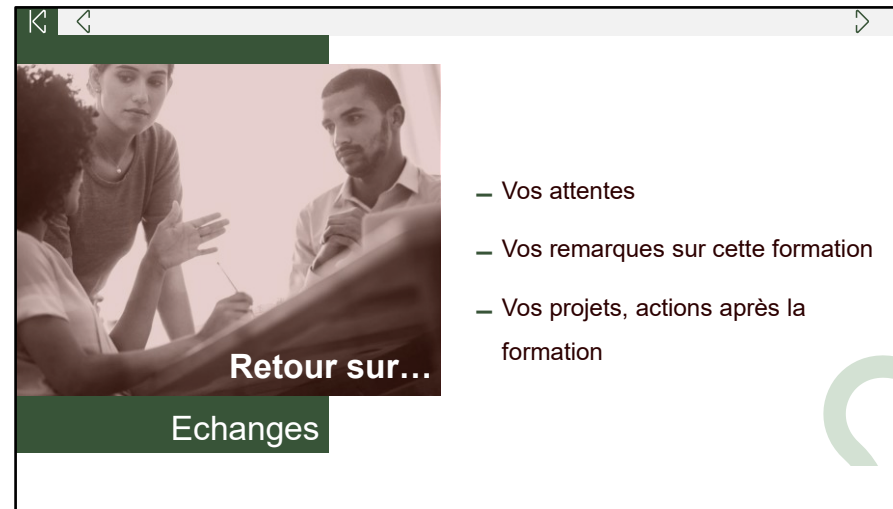
## Les index

Un *index* permet **d'améliorer le temps d'exécution des requêtes d'interrogation de données**

***CREATE INDEX :***  
*CREATE [UNIQUE] INDEX nom\_index ON table (col1, col2,...);*

***DROP INDEX :***  
*DROP INDEX nom\_index [ON table];*

Je reste disponible pour toute question



The slide features a dark green header bar at the top with navigation icons (back, forward, and search). Below the header, on the left, is a photograph of three people (two women and one man) in a meeting, with the text "Retour sur..." overlaid. To the right of the photo is a list of topics. At the bottom left, the word "Echanges" is written in white on a dark green background. A large, light green curved arrow graphic is positioned on the right side of the slide.

- Vos attentes
- Vos remarques sur cette formation
- Vos projets, actions après la formation

**Retour sur...**

**Echanges**

