

# Introduction aux bases de données

La commande CREATE TABLE  
La commande INSERT INTO

# Les commandes: CREATE TABLE INSERT INTO

## Plan de la séance

- Retour sur la dernière séance:
  - Point de vue de l'étudiant
  - Point de vue de l'enseignant.
- Rappel: définition d'une table
- La commande CREATE TABLE
  - Définitions
  - Types de données Oracle
  - Les contraintes d'intégrités
  - Syntaxe
  - Exemples, de création de table
  - La commande INSERT INTO
  - Laboratoire 2

# Rappels

- Une table est un ensemble de lignes et de colonnes.
- Une table est l'objet dans lequel les données sont stockées.  
C'est l'objet principal manipulé par un SGBDR
- Les lignes sont aussi appelées : Enregistrements
- Les colonnes sont aussi appelées: Attributs

# La commande CREATE TABLE

## Introduction:

- Tous les objets de la base de données sont créés avec la commande CREATE. Cette commande est une commande DDL. (Data Definition Language)
- La commande CREATE TABLE permet de créer une table dans une base de données.
- Pour créer une table nous avons besoin de connaître:
  - La liste des attributs de la table
  - Le type de données pour chaque attribut
  - Les contraintes d'intégrité, s'il y'en a, pour chaque attribut
  - Et l'ensemble des contraintes d'intégrité sur la table, s'il y'en a.
- Nous devons connaître aussi le nom de la table qui sera créée.

# Les types de données Oracle .

- Type de données: Dans tous les SGBDs (comme pour les langages de programmation) les données qui seront stockées ont un type. Pour Oracle, voici les principaux types

Pour le reste des types de données, allez sur: [https://docs.oracle.com/cd/B28359\\_01/server.111/b28318/datatype.htm#CNCPT183](https://docs.oracle.com/cd/B28359_01/server.111/b28318/datatype.htm#CNCPT183)

Types	Significations , Exemples
VARCHAR2(n)	Chaîne de caractères de longueur variable. La taille maximale de cette chaîne est déterminée par la valeur <b>n</b> et peut atteindre 4000 caractères (bytes). La longueur minimale est 1. la précision du n est obligatoire. Exemple : NomProgramme VARCHAR2(20)
CHAR(n)	Chaîne de caractères de longueur fixe allant de 1 à 2000 caractères (bytes). La chaîne est complétée par des espace si elle est plus petite que la taille déclarée Exemple CodeProgramme CHAR(3).
NUMBER(n,d)	Pour déclarer un nombre sur maximum <b>n</b> chiffres (positions) dont <b>d</b> chiffres (positions) sont réservés à la décimale. <b>n</b> peut aller de 1 à 38. Exemple salaire NUMBER(6,2). La valeur maximale du salaire dans ce cas est 9999,99
DATE	Donnée de type date située dans une plage comprise entre le 1er janvier 4712 av JC et le 31 décembre 9999 ap JC stockant l'année, mois, jour, heures, minutes et secondes
BLOB	Binary Large Object : Gros objet binaire pouvant aller jusqu'à 4 gigaoctet : Exemple photo BLOB

# Définitions, les contraintes d'intégrité.

- Une contrainte d'intégrité est une règle sur la table qui permet d'assurer que les données stockées dans la base de données soient cohérentes par rapport à leur signification. Avec Oracle, on peut implémenter plusieurs contraintes d'intégrité.

Contraintes	Significations , Exemples
PRIMARY KEY	<p>La contrainte de PRIMARY KEY, permet de définir une clé primaire pour la table. Une clé primaire est un attribut qui permet d'identifier chaque occurrence(enregistrement) d'une table de manière unique. Ou encore c'est un attribut permettant d'identifier chaque ligne d'une table d'une manière unique.</p> <p><b>Exemples :</b> Le numéro d'admission d'un étudiant Un numéro de facture</p> <p>Le nom de l'étudiant ne peut être défini comme Clé primaire.</p>
CHECK	<p>Indique les valeurs permises qui peuvent être saisies pour la colonne (attribut) lors de l'entrée des données ou une condition à laquelle doit répondre une valeur insérée. La condition doit impliquer le nom d'au moins une colonne. Les opérateurs arithmétiques (+, *, /, -), les opérateurs de comparaisons et les opérateurs logiques sont permis.</p> <p>Exemples: La note de l'étudiant est comprise entre 0 et 100. Le salaire minimum est plus grand que 15.</p>

# Les contraintes d'intégrité.

## Contraintes d'intégrité (suite)

Contraintes	Significations , Exemples
NOT NULL	Indique que la valeur de la colonne ou de l'attribut est obligatoire. Si cette contrainte n'est pas précisée alors par défaut la valeur est NULL. Exemple: le nom de l'étudiant est obligatoire
UNIQUE	Indique que les valeurs saisies pour les colonnes (attributs) doivent être uniques, ce qui veut dire pas de doublons. Exemple: Le numéro d'assurance sociale, le numéro d'un permis de conduire, un courriel. Le nom de l'étudiant ne peut pas avoir cette contrainte.
DEFAULT	Indique la valeur par défaut que prendra l'attribut si aucune valeur n'est saisie.

La contrainte de PRIMARY KEY a les contraintes UNIQUE et NOT NULL

# Syntaxe simplifiée

## Syntaxe simplifiée : CREATE TABLE

```
CREATE TABLE nom_table  
(  
  nom_colonne type_donnee_colonne [definition_contrainte_colonne],  
  nom_colonne type_donnee_colonne [definition_contrainte_colonne],  
  ....  
  [definition_contrainte_table],  
  ....  
  [definition_contrainte_table]  
);
```



# Exemples

## Exemple1

```
CREATE TABLE Etudiants
(
  numad NUMBER(10,0) CONSTRAINT pk_etudiant PRIMARY KEY,
  nom VARCHAR2(20) NOT NULL,
  prenom VARCHAR2 (20)
);
```

### Lecture de l'exemple:

- Le nom de la table est Etudiants
- Chaque définition de colonne se termine par une virgule sauf la dernière
- Le numad a la contrainte de PRIMARY KEY, cette contrainte est sur la colonne numad
- Pour définir une contrainte, on utilise le mot réservé CONSTRAINT . Le nom pk\_etudiant représente le nom de la contrainte dans le SGBD

# Exemples

## Exemple 2

```
CREATE TABLE Employes
(
empno NUMBER(4,0) CONSTRAINT pk_employes PRIMARY KEY,
nom VARCHAR2(20) NOT NULL,
prenom VARCHAR2 (20),
salaire number(8,2) CONSTRAINT ck_salaire CHECK(salaire> 50000)
);
```

### Lecture de l'exemple:

- Le nom de la table est Employes
- Le empno a la contrainte de PRIMARY KEY.
- Pour définir une contrainte, on utilise le mot réservé CONSTRAINT . Le nom pk\_employes représente le nom de la contrainte dans le SGBD
- Il y a une contrainte NOT NULL sur le nom
- Il y a une contrainte CHECK sur le salaire. Cette contrainte est définie avec le mot réservé **CONSTRAINT**. Le nom de la contrainte est ck\_salaire

# Exemples

## Exemple 3, cette écriture est fortement déconseillée. (et à ne pas utiliser)

```
CREATE TABLE EmployesBidon
(
empno NUMBER(4,0) PRIMARY KEY,
nom VARCHAR2(20) NOT NULL,
prenom VARCHAR2 (20),
salaire number(8,2) CHECK(salaire> 50000)
);
```

### Lecture de l'exemple:

- Le nom de la table est EmployesBidon
- Le empno a la contrainte de PRIMARY KEY. Nous n'avons pas donné de nom à cette contrainte
- Le salaire a une contrainte de CHECK. Nous n'avons pas donné de nom à cette contrainte
- Lorsque vous ne donnez pas de nom aux contraintes, le système se chargera de leur donner un nom.
- L'exemple est ici uniquement parce que vous allez le trouver dans stack overflow. Cette définition des contraintes est déconseillée.

# Exemples

## Exemple 4

```
CREATE TABLE personnes
(
numero NUMBER(4,0) CONSTRAINT pk_personne PRIMARY KEY,
nom VARCHAR2 (15) NOT NULL,
prenom VARCHAR2 (15),
courriel VARCHAR2(40) UNIQUE,
ville VARCHAR2 (20) DEFAULT 'Montréal' CONSTRAINT ck_ville CHECK(ville IN ('Montréal','Laval','Québec'))
);
```

Nous avons:

- Une contrainte de PRIMARY KEY sur la colonne numero. Cette contrainte a un nom : pk\_personne
- Une contrainte NOT NULL sur le nom. Cette contrainte a un nom donné par le système
- Une contrainte UNIQUE sur le courriel. Cette contrainte a un nom donné par le système
- Une contrainte CHECK sur la ville. Cette contrainte a un nom : ck\_ville.
- Lorsque la ville n'est pas saisie, par défaut la valeur est Montréal.

# CREATE TABLE: Important

## Important:

- Les tables sont des objets de la base de données. Les noms doivent être uniques. Vous ne pouvez pas avoir deux tables Employes dans votre BD
- Les noms des tables doivent être significatifs.
- Les noms des tables ne doivent pas être des mots réservés du SGBD. Exemple de mot réservés: Table, sequence, sysdate, create, cycle, constraint, primary etc...
- Les colonnes des tables doivent avoir des noms significatifs et ne doivent pas être des mots réservés.
- Les noms de colonnes sont uniques dans une table, mais pas dans la base de données.
- Les contraintes de PRIMARY KEY et CHECK doivent être définies avec le mot réservé CONSTRAINT et doivent avoir un nom significatif.
- Les contraintes sont des objets de la base de données, le nom doit être unique.
- Les contraintes de NOT NULL et UNIQUE sont généralement définies sans nom.( Le système se charge de leur donner un nom)
- Pour tous les exemples précédents, nous avons défini les contraintes lors de la définition des colonnes. Ce sont des contraintes niveau colonnes. (ou attribut)

# Exemples

## Exemple 5

```
CREATE TABLE EmployesClg
(
empno NUMBER(4,0) ,
nom VARCHAR2(20) NOT NULL,
prenom VARCHAR2 (20),
salaire number(8,2),
CONSTRAINT pk_employes PRIMARY KEY (empno),
CONSTRAINT ck_salaire CHECK(salaire> 50000)
);
```

### Lecture de l'exemple:

- La contrainte de PRIMARY KEY est définie comme si c'était une colonne. Remarquez la virgule jaune. C'est ce que nous appelons une définition de contrainte au niveau TABLE.
- La contrainte de CHECK est définie comme si c'était une colonne. Elle est définie au niveau TABLE.

# La commande INSERT INTO

## Définition et syntaxe

- Cette commande permet d'insérer des données dans une table, une ligne à la fois. C'est une commande du DML (Data Manipulation Language)
- Deux syntaxes sont possibles pour cette commande

Syntaxe 1, cette syntaxe indique que l'on doit fournir les valeurs valides pour toutes les colonnes de la table

```
INSERT INTO <nom_de_table> VALUES (<liste de valeurs>);
```

Exemple: pour la table de l'exemple 5

```
INSERT INTO Employes values(1,'Leroy','Rémi', 60000);
```

# La commande INSERT INTO

## Définition et syntaxe

Syntaxe2 , cette syntaxe indique que l'on doit fournir les valeurs valides pour chaque colonne indiquée dans la commande INSERT INTO

```
INSERT INTO <nom_de_table>(<nom_de_colonne>) VALUES (<liste_de_valeurs>);
```

Exemple: pour la table de l'exemple 5

```
INSERT INTO Employes (empno, nom, salaire) values(1,'Leroy', 60000);
```



# INSERT INTO, Important

## Important:

- La commande INSERT INTO insère une ligne à la fois.
- Aucune insertion n'est possible si les contraintes d'intégrités ne sont pas respectées.

INSERT INTO Employes (nom, salaire) values (Leroy', 60000); ne marchera pas car il n' y a pas de empno . Le empno est obligatoire (NOT NULL).

INSERT INTO Employes (empno, nom, salaire) values(1,'Leroy', 60000); va marcher

INSERT INTO Employes (empno, nom, salaire) values(1,'Lesinge', 80000); Ne marchera pas, car empno doit-être unique

INSERT INTO Employes (empno, nom, salaire) values(2,'Lesinge', 20000); Ne marchera pas, car le salaire est plus petit que 50000

# INSERT INTO, Important

## Important:

- Une valeur de type caractère (CHAR ou VARCHAR2) doit être mise entre apostrophes. Si la chaîne de caractère contient des apostrophes, ceux-ci doivent être doublés.

```
INSERT INTO employes VALUES (3,'O''Brian','Kévin',100000);
```

- Le type numérique (NUMBER) est saisi en notation standard. La virgule décimale est remplacée par un point lors de la saisie

```
INSERT INTO employes VALUES (4,'O''Neil','Carl',800000.88)
```

- Le type date doit être saisi selon la norme américaine (JJ-MMM-AA pour 12 jan 21) et entre apostrophes. Pour saisir une date dans n'importe quel format, il faut s'assurer de la convertir dans le format avec la fonction TO\_DATE

Si vous ne connaissez pas le format de la date, exécutez: **SELECT SYSDATE FROM DUAL;**

- Lorsque la valeur d'une colonne n'est pas connue et que celle-ci possède une contrainte de NOT NULL, alors on peut saisir le NULL entre apostrophe comme valeur pour cette colonne.

# INSERT INTO, Important

## Important:

- Il est possible d'utiliser une expression arithmétique dans la commande INSERT INTO à condition que cette colonne soit de type numérique.

```
INSERT INTO employes VALUES(5,'Patoche','Alain',(50000 +(50000*0.1)/2));
```

- Si des valeurs dans certaines colonnes ne doivent pas être saisies (contiennent des valeurs par défaut) alors la précision des colonnes dans lesquelles la saisie doit s'effectuer est obligatoire. Noter que les valeurs à saisir doivent être dans le même ordre de la spécification des colonnes. Voir syntaxe 2

Table personnes de l'exemple 4 (dans la colonne Villes, Montréal sera inséré par défaut)

```
INSERT INTO personnes(numero,nom,prenom,courriel) VALUES(11,'Saturne','Lune','Saturne@gmail.com');
```

# INSERT INTO, Important

## **Important:**

- Il est possible d'utiliser des insertions à partir d'une table existante (commande SELECT et une sous-requête -----à voir plus loin)
- Il est possible d'utiliser une séquence pour l'insertion automatique d'un numéro séquentiel pour une colonne (à voir plus loin)
- Après les insertions, il est recommandé d'exécuter la commande COMMIT pour enregistrer vos données (à voir plus loin)

# INSERT INTO, Conclusion

## Conclusion

- INSERT INTO est une commande DML qui permet d'insérer des données dans une table. Une ligne à la fois.
- La commande INSERT INTO ne pourra pas s'exécuter si les contraintes d'intégrités ne sont pas respectées. En tout temps, il faut vérifier les contraintes d'intégrité.
- Pour que les insertions (les données) sont définitivement enregistrées dans la BD, il suffit d'exécuter un COMMIT.

# Les commandes CREATE TABLE INSERT INTO



Conclusion



Questions