# 1) INSERTION / MISE A JOUR / SUPPRESSION DE DONNEES

## a) Insertion de nouvelles données dans une table (INSERT INTO)

Pour insérer des données dans une base, il y a 2 syntaxes principales :

- Insérer une ligne en indiquant les informations pour chaque colonne existante (en respectant l'ordre)
- Insérer une ligne en spécifiant les colonnes que vous souhaitez compléter. Il est possible d'insérer une ligne renseignant seulement une partie des colonnes

### ➢ Insérer une ligne en spécifiant toutes les colonnes

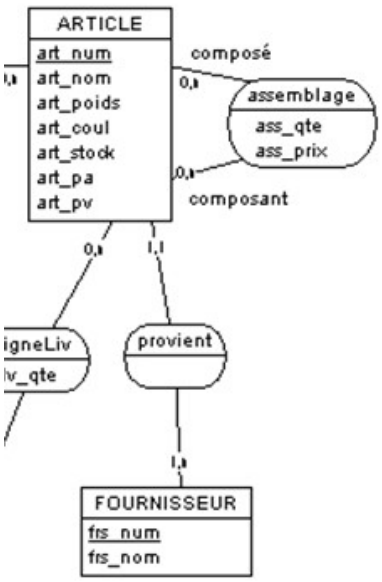
La syntaxe pour remplir une ligne avec cette méthode est la suivante :

**INSERT INTO** <table>
**VALUES** (<valeur1>,<valeur2>,…,<valeur n>)

Cette syntaxe possède les avantages et inconvénients suivants :

- Obliger de remplir toutes les données, tout en respectant l'ordre des colonnes
- L'ordre des colonnes doit rester identique sinon certaines valeurs prennent le risque d'être complétée dans la mauvaise colonne

**Remarque** : Le SGBD se réserve le droit de ne pas autoriser une insertion si celle-ci va à l'encontre d'une contrainte posée sur la base (clé primaire, clé étrangère, …).




**ARTICLE** (art_num, art_nom, art_poids, art_coul, art_stock, art_pa, art_pv, art_frs)
Contrainte de clé primaire : art_num
Contrainte de clé étrangère : art_frs en référence à FOURNISSEUR

Exemple : Insérer un nouvel article dont voici les caractéristiques : référence A16, PACK CRAYON, 100 grammes, TRANSPARENT, 50 unités en stock, prix d'achat 3 €, prix de vente 5 €, fournisseur F02.

INSERT INTO ARTICLE (art_num, art_nom, art_poids, art_coul, art_stock, art_pa, art_pv, art_frs)
VALUES ('A16', 'PACK CRAYON', 100, 'TRANSPARENT', 50, 3, 5, 'F02')

- Insérer une ligne en spécifiant seulement les colonnes souhaitées

Cette deuxième solution est très similaire, excepté qu'il faut indiquer le nom des colonnes avant "VALUES". La syntaxe est la suivante :

```
INSERT INTO table (nom_colonne_1, nom_colonne_2, ...)
VALUES ('valeur 1', 'valeur 2', ...)
```

**Remarque :** il est possible de ne pas renseigner toutes les colonnes. De plus, l'ordre des colonnes n'est pas important.

Exemple : Insérer un nouvel article dont voici les caractéristiques : référence A17, PACK STYLOS, ROUGE, 40 unités en stock, prix d'achat 3 €, fournisseur F02.

INSERT INTO ARTICLE (art_num, art_nom, art_coul, art_stock, art_pa, art_frs)
VALUES ('A17', 'PACK STYLOS', 'ROUGE ', 40, 3 , 'F02')

- Insertion de plusieurs lignes à la fois

Il est possible d'ajouter plusieurs lignes avec une seule requête. Pour ce faire, il convient d'utiliser la syntaxe suivante :

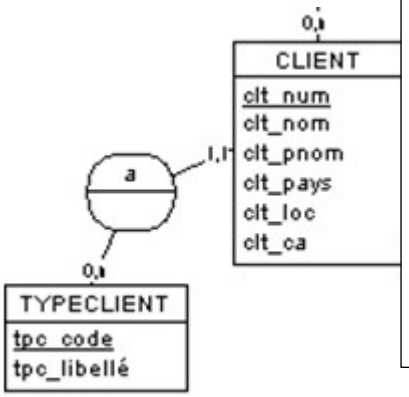
```
INSERT INTO client (prenom, nom, ville, age)
VALUES
('Rébecca', 'Armand', 'Saint-Didier-des-Bois', 24),
('Aimée', 'Hebert', 'Marigny-le-Châtel', 36),
('Marielle', 'Ribeiro', 'Maillères', 27),
('Hilaire', 'Savary', 'Conie-Molitard', 58);
```

Un tel exemple sur une table vide va créer le tableau suivant :

| id | prenom | nom | ville | age |
|---|---|---|---|---|
| 1 | Rébecca | Armand | Saint-Didier-des-Bois | 24 |
| 2 | Aimée | Hebert | Marigny-le-Châtel | 36 |
| 3 | Marielle | Ribeiro | Maillères | 27 |
| 4 | Hilaire | Savary | Conie-Molitard | 58 |

b) **Mise à jour d'enregistrements (UPDATE)**

**UPDATE** <table>
**SET** <champ> = <nouvelle valeur>
**WHERE** <critère>




**TYPECLIENT** (tpc_code, tpc_libellé)
Contrainte de clé primaire : tpc_code

**CLIENT** (clt_num, clt_nom, clt_pnom, clt_pays, clt_loc, clt_ca, clt_type)
Contrainte de clé primaire : clt_num
Contrainte de clé étrangère : clt_type en référence à TYPECLIENT

Exemples :

- Le client DEFEEZ a déménagé pour BREST

  UPDATE CLIENT
  SET clt_loc = ’BREST’
  WHERE clt_nom = ‘DEFEEZ’

- Ajuster le prix de vente de tous les articles dont le poids est inférieur à 100 grammes, de telle façon que le prix de vente soit le double du prix d'achat.

ARTICLE (art_num, art_nom, art_poids, art_coul, art_stock, art_pa, art_pv, art_frs)

UPDATE ARTICLE
SET art_pv = 2 * art_pa
WHERE art_poids < 100

## c ) Suppression d'enregistrements (DELETE)

**DELETE FROM** <table>
**WHERE** <critère>

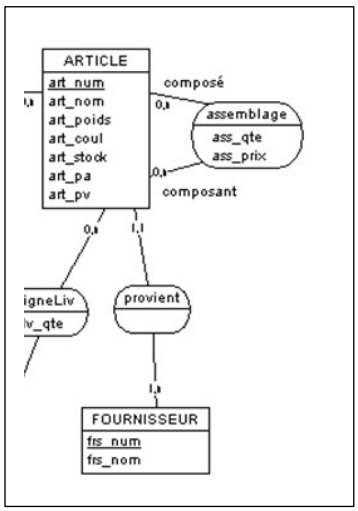
- Supprimer le client C09 car il n'a jamais commandé d'article !

DELETE FROM CLIENT
WHERE clt_num = 'C09'

- Supprimer le fournisseur F01. Que se passe-t-il ?

DELETE FROM FOURNISSEUR
WHERE frs_num = 'F01'

S'il y a des articles qui proviennent de ce Fournisseur la suppression n'est pas possible.



## 2) LDD (Langage de Définition des Données)

### ➢ Création d'une table

```
CREATE TABLE <nom de la table>
( <attribut 1> <type attribut 1>,
        ...
  <attribut n> <type attribut n>)
```

a) Les types de données les plus courants :

Numériques :

- INTEGER : entier long,
- SMALLINT : entier court,
- NUMERIC(p,e) : p(précision) e(échelle), représente des nombres non entiers, mais dont la valeur est exacte (par exemple une valeur monétaire déclarée prix NUMERIC(5,2) pourra stocker jusqu'à 99999,99 €),
- DECIMAL(p,e), idem NUMERIC,
- REAL : réel, stocke une valeur approchée avec une précision propre à la machine,
- DOUBLE PRECISION : idem REAL avec une précision plus grande,
- FLOAT(p) : réel, stocke une valeur approchée avec une précision définie par un paramètre,

Chaînes de caractères :

- CHAR : caractère unique,
- CHAR(n) : chaîne de caractères de longueur n, l'espace non utilisé est rempli avec des blancs, à utiliser pour les clés,
- VARCHAR(n) : idem CHAR mais seul le nombre exact de caractères est stocké, ce type permet de gagner de la place,
- NCHAR : représente le type CHAR avec le jeu de caractères national tel que le français, l'espagnol…,
- NVARCHAR(n) : idem VARCHAR avec le jeu de caractère national,

Types temporels :

- DATE : date, sans paramètre sous la forme 'MMJJAAAA',
- TIME(n) : sous la forme 'HHMMSS', le paramètre n permet de spécifier le nombre de chiffres après la virgule dans les secondes (0 par défaut),
- SMALLDATETIME : Date abrégée,

**Remarque** : les différents SGBD possèdent des types spécifiques.

b) Spécifier une contrainte

☞Contrainte de CLE PRIMAIRE (contrainte d'intégrité d'entité)

Afin d'assurer l'intégrité de la base, une contrainte de clé primaire est chargée de vérifier qu'il n'y aura pas de doublons dans le champ identifiant.

Cas d'une clé primaire mono-attribut (utilisation d'une contrainte sur colonne)

**CREATE TABLE** <nom de la table>
( <attribut clé> <type attribut 1> **PRIMARY KEY**
...
<attribut n> <type attribut n> )

**MAGASIN** (id, ville, gerant)
Contrainte de clé primaire : id

- Créer la table "magasin"

```
CREATE TABLE MAGASIN (
    id integer PRIMARY KEY,
    ville varchar(50),
    gérant varchar(50))
```

Cas d'une clé primaire multi-attributs (utilisation d'une contrainte sur table)

**CREATE TABLE** <nom de la table>
( <attribut clé 1> <type attribut 1>
<attribut clé 2> <type attribut 2>
...
<attribut n> <type attribut n>
**PRIMARY KEY** (<attribut clé 1>,<attribut clé 2> )

- Créer la table 'ligneCmd"

LIGNECMD (id_commande, id_article, qte, qt_livree, dateliv)
Contrainte de clé primaire : id_commande, id_article
Contrainte de clé étrangère : id_commande en référence à id de la table COMMANDE
Id_article en référence à id de la table ARTICLE

```
CREATE TABLE LIGNECMD (
        id_commande integer,
         id_article integer,
        qte integer,
        qt_livree integer,
        dateliv DATE,
        PRIMARY KEY (id_commande, id_article)
        ……
```

☞Contrainte de CLE ETRANGERE (contrainte d'intégrité référentielle)

Afin d'assurer l'intégrité de la base, une contrainte de clé étrangère est chargée de vérifier si la valeur de la table source est bien présente en tant que clé dans la table cible.

Version contrainte sur colonne

**CREATE TABLE** <nom de la table>
( <attribut 1> <type attribut 1> ...
...
<attribut n> <type attribut n> **REFERENCES** <nom de la table cible>(attribut))

ou version contrainte sur table

**CREATE TABLE** <nom de la table>
( <attribut 1> <type attribut 1> ...
...
<attribut n> <type attribut n>,
**FOREIGN KEY** <attribut n> **REFERENCES** <nom de la table cible>(attribut))

**Attention** : En SQL dans le cas d'une clé primaire multi-attributs, en plus de la contrainte PRIMARY KEY, il faut également préciser l'intégrité référentielle REFERENCES !

- Corriger la création de la table ligneCmd en tenant compte de l'intégrité référentielle avec commande (on ne doit pas pouvoir créer des lignes d'une commande qui n'existe pas) et avec ARTICLE.

```
CREATE TABLE LIGNECMD (
id_commande integer,
 id_article integer,
qte integer,
qt_livree integer,
dateliv DATE,
PRIMARY KEY (id_commande, id_article)
FOREIGN KEY id_commande REFERENCES COMMANDE(id),
FOREIGN KEY id_article REFERENCES ARTICLE(id))
```

☞Contraintes de champ non vide (contrainte de domaine)

Si rien n'est précisé un champ peut ne pas être renseigné, dans ce cas il est affecté de la valeur NULL. Dans le cas contraire, il faut préciser la clause NOT NULL :

| <champ> <type champ> **NOT NULL** |
|---|

☞Contraintes de valeur par défaut (contrainte de domaine)

Il est possible de préciser une valeur par défaut avec la clause DEFAULT :

| <champ> <type champ> **DEFAULT** <valeur par défaut> |
|---|

☞Contraintes de vérification d'intégrité

La clause CHECK est précisée au niveau du CREATE TABLE et permet de vérifier lors de l'insertion ou de la modification d'un enregistrement une contrainte définie par le programmeur. La clause CHECK peut vérifier une contrainte sur une colonne ou sur l'enregistrement complet.

Voici quelques exemples :

**Exemple 1 (contrainte sur colonne)** : Vérifier que l'affectation d'un champ se fasse dans une liste de valeurs :

clt_type CHAR(16) **CONSTRAINT** typeDuClient **CHECK** (clt_type **IN** ('Particulier', Administration', 'Grand Compte', 'PME'))

**Exemple 2 (contrainte sur table)** : Dans la table assemblage, un produit composé n'est pas composé de lui_même !

ass_composé CHAR(8),
ass_composant
…
**CONSTRAINT** PasLuiMeme **CHECK** (ass_composé != ass_composant)

**Exemple 3 (contrainte sur table)**: La quantité livrée ne peut être supérieure à la quantité commandée :

…
**CONSTRAINT** PasSupérieur **CHECK** (lcd_liv<=lcd_qte)

- **Suppression d'une table**

**DROP TABLE** <nom de la table>

- **Modification de la structure d'une table**

a) Ajout d'un attribut

```
ALTER TABLE <nom de la table>
ADD COLUMN <attribut> <type>, <attribut><type>
```

b) Agrandir la taille d'un attribut

```
ALTER TABLE <nom de la table>
ALTER COLUMN <attribut> <type>(nouvelle taille)
```

c) Supprimer un attribut

```
ALTER TABLE <nom de la table>
DROP COLUMN <attribut>
```

d) Ajouter une contrainte sur table

```
ALTER TABLE <nom de la table>
ADD CONSTRAINT <attribut>
```