# Ordres DML

Les ordres DML (Data Manipulation Language, langage de manipulation des données), permettent

* l’ajout de nouvelles lignes dans une table
* la modification des valeurs de certaines colonnes des lignes d’une table, selon un critère de sélection des lignes à modifier
* la suppression de lignes d’une table, selon un critère de sélection des lignes à supprimer

Les tables suivantes seront utilisées pour les exemples :

* **CLIENT (idCLi, nomCli, telCli)**, idCLi est clef primaire
* **COMMANDE (idCom, dateCom, idCli, idProd, delai)**, idCom est clef primaire, idCli est clef étrangère vers Client (idCli), idProd est clef étrangère vers Produit (idProd)
* **PRODUIT (idProd, designProd, prixProd)**, idProd est clef primaire
* **HISTOCOMMANDE (idCom, dateCom, idCli,idProd)**

# INSERT INTO

L’ordre INSERT INTO permet l’ajout d’une ou plusieurs lignes dans une table

* à partir de données fixes fournies
* ou à partir du résultat d’une requête SELECT.

## Ajouter une ou plusieurs lignes à partir de données fixes

**INSERT INTO nomDeLaTable**

**(listeDesColonnes)**

**VALUES (listeDesValeurs) ;**

où :

* **nomDeLaTable** est le nom de la table dans laquelle les lignes seront ajoutées
* **listeDesColonnes** est une liste de noms de colonnes de cette table, séparées par des virgules (*toutes les colonnes de la table ou celles qui sont obligatoires obligatoires*) ; si cette liste n’est pas indiquée, toutes les colonnes de la tables devront trouver une valeur dans la liste ci-dessous
* **listeDesValeurs** est la liste des valeurs que chacune des colonnes va prendre pour la ligne qui va être ajoutée, séparées par des virgules (*la liste des valeurs est dans le même ordre que celle des colonnes concernées*)

Par exemple, pour ajouter le client de numéro 10, société Au bout du Monde, téléphone 01.22.33.44.88 :

**INSERT INTO CLIENT**

**(idCli, nomCli, telCli)**

**VALUES**

**(10,"Au bout du Monde","01.22.33.44.88");**

Il est possible d’insérer plusieurs lignes de données dans un seul ordre INSERT INTO :

**INSERT INTO CLIENT**

**(idCli, nomCli, telCli)**

**VALUES**

**(25,"Au bon marché","01.15.16.17.17 poste 16"),**

**( 8,"Chez Maurice","");** (point-virgule pour terminer)

## Ajouter un ou plusieurs lignes à partir du résultat d’une requête

**INSERT INTO nomDeLaTable**

**(listeDesColonnes)**

**ordreSelect ;**

où :

* **nomDeLaTable** est le nom de la table dans laquelle les lignes seront ajoutées
* **listeDesColonnes** est une liste de noms de colonnes de cette table, séparées par des virgules (*toutes les colonnes de la table ou celles obligatoires*)
* **orSelect** est une requête SQL SELECT qui retourne la liste des valeurs que chacune des colonnes va prendre pour la ligne qui va être ajoutée, séparées par des virgules (*la liste des valeurs et dans le même ordre que celle des colonnes concernées*) ; l’ordre SELECT peut comporter des jointures, etc.

Par exemple, pour ajouter des commandes à l’historique des commandes, par exemple celles qui ont été passées avant le 01 janvier 2016:

**INSERT INTO HISTOCOMMANDE**

**(idCom, dateCom, idCli, idProd)**

**SELECT**

**idCom, dateCom, idCli, idProd**

**FROM COMMANDE**

**WHERE dateCom < "2016-01-01";**

(Remarquez qu’on n’a pas repris le délai dans l’historique).

# UPDATE

L’ordre UPDATE permet la modification d’une ou plusieurs valeurs de colonnes de plusieurs lignes sélectionnées dans une table (sans sélection, toutes les lignes sont modifiées

**UPDATE nomDeLaTable**

**SET colonne1 = nouvelleValeur1 ,colonne2 = nouvelleValeur2, etc.**

**WHERE critère;**

où :

* **nomDeLaTable** est le nom de la table pour laquelle des valeurs de colonnes seront modifiées
* **colonne1, colonne2, …** : noms des colonnes qui seront modifiées
* **nouvelleValeur1, nouvelleValeur2, … :** nouvelles valeurs pour chacune des colonnes (remplace l’ancienne)
* **critère :** critère de sélection des lignes qui seront modifiées (le critère peut utiliser des requêtes imbriquées)

Par exemple, pour modifier le téléphone du client 10, et mettre 01.22.33.44.55:

**UPDATE CLIENT**

**SET telCli = "01.22.33.44.55"**

**WHERE idCli = 10 ;**

Par exemple, pour modifier la désignation et le prix d’un produit de numéro 12 :

**UPDATE PRODUIT**

**SET designProd = "cahier de dessin", prixProd = 1.50**

**WHERE idProd = 12 ;**

Par exemple, pour baisser les prix des produits sans commande de 10%:

**UPDATE PRODUIT**

**SET prixProd = prixProd \* 0.90**

**WHERE idProd NOT IN (SELECT DISTINCT idProd FROM COMMANDE) ; ;**

ATTENTION, l’ordre SQL suivant modifie le prix de tous les produits (ancien prix augmenté de 10%), car il n’y a pas de clause WHERE pour choisir des lignes (ici, c’est ce qu’on souhaite faire…):

**UPDATE PRODUIT**

**SET prixProd = prixProd \* 1.10 ;**

# DELETE FROM

L’ordre DELETE FROM permet la suppression d’une ou plusieurs lignes sélectionnées dans une table (sans sélection, toutes les lignes sont supprimées !)

**DELETE FROM nomDeLaTable**

**WHERE critère;**

où :

* **nomDeLaTable** est le nom de la table à partir de laquelle des lignes seront supprimées
* **critère :** critère de sélection des lignes qui seront supprimées (le critère peut utiliser des requêtes imbriquées)

Par exemple, pour supprimer les commandes antérieures au 1er janvier 2016 (elles ont été ajoutées à l’historique des commandes)

**DELETE FROM COMMANDE**

**WHERE dateCom < "2016-01-01";**

Par exemple, pour supprimer les produits pour lesquels aucune commande n’existe :

**DELETE FROM PRODUIT**

**WHERE idProd NOT IN (SELECT DISTINCT idProd FROM COMMANDE);**

ATTENTION, l’ordre SQL suivant supprime tous les produits, car il n’y a pas de clause WHERE pour choisir des lignes

**DELETE FROM PRODUIT;**

Par chance, les contrôles d’intégrité référentielle (de idProd de COMMANDE vers idProd de PRODUIT) vont garantir qu’on ne peut supprimer un produit si une commande y fait référence (***à moins que l’informaticien l’ait autorisé…)***

**PRUDENCE DONC SUR LA SUPPRESSION DE LIGNES…(et LA MODIFICATION)**

**SANS CLAUSE 'WHERE'**