



BASES DE DONNÉES :

COMPLÉMENT DML ET DDL

Bases de données: Avancé
420-465-CH

Sana El Bahloul
Hiver 2018

Partie 1:

DML (Langage de manipulation de données) :

- Ajouter, Modifier et Supprimer des données

Partie2:

DDL (Langage de définition de données)

- Créer une BD, ses table et ses colonnes



LE DML C'EST PLUS QUE SELECT

- Le DML – Data Manipulation Language consiste en l'ensemble des commandes permettant de **consulter** et de **modifier** les données d'une BD.
- Soit:
 - SELECT – Consultation des données
 - INSERT - Ajout de données
 - UPDATE - Modification de données
 - DELETE - Suppression de données



AJOUT DE DONNÉES

- La commande permettant d'ajouter une ligne dans une table est **INSERT**.

```
INSERT INTO tblNomTable(liste des colonnes)  
      VALUES(liste des valeurs);
```

- On effectue une commande **INSERT** pour chaque ligne à ajouter.

```
INSERT INTO tblPersonne(nomPers, prenPers)  
      VALUES('Gagnon','Jean');
```



RÈGLES POUR L'AJOUT

- Si certaines valeurs sont omises, alors les valeurs par défaut définies lors de la création seront utilisées.
- S'il n'y a pas de valeurs par défaut et que NULL est permis, la donnée contiendra NULL.
- S'il n'y a pas de valeurs par défaut et que le champs a la contrainte NOT NULL, la valeur sera:

Valeur	Type donnée
0	pour un nombre
' '	pour une chaîne
0000-00-00	pour une date
00:00:00	pour une heure



AJOUT SANS NOMS DE COLONNES

- Une syntaxe plus courte mais avec laquelle toutes les valeurs doivent être fournies.

INSERT INTO tblNomTable **VALUES**(valeurs dans l'ordre)

- Soit la définition de table suivante:

```
CREATE TABLE tblBallon
( tailleBallon INT NOT NULL PRIMARY KEY,
  coulBallon   VARCHAR(20)
);
```

```
INSERT INTO tblBallon VALUES(20, 'rouge') bon
INSERT INTO tblBallon VALUES('rouge', 20) mauvais
INSERT INTO tblBallon VALUES('rouge' ) mauvais
```

AJOUTER PLUSIEURS ENREGISTREMENTS

- Il est possible d'ajouter plusieurs enregistrements dans la même commande INSERT

```
INSERT INTO tblNomTable VALUES(liste de valeurs dans l'ordre),  
                                (liste d'autres valeurs),(liste d'encore d'autres valeurs);
```

- Avec ou sans les noms de colonnes

```
INSERT INTO BALLON(taille, couleur) VALUES(20, 'rouge'),  
                                           (35, 'vert fluo'), (17, 'orange');
```

```
INSERT INTO tblBallon VALUES (20, 'rouge'), (35, 'vert fluo'),  
                               (17, 'orange');
```

- Ou dans des commandes séparées

```
INSERT INTO tblBallon VALUES (20, 'rouge');  
INSERT INTO tblBallon VALUES (35, 'vert fluo');  
INSERT INTO tblBallon VALUES (17, 'orange');
```

MODIFICATION DE DONNÉES

- La commande permettant de modifier des données dans une table est **UPDATE**.

```
UPDATE tblNomTable SET champ=valeur, ...  
[ WHERE condition ];
```

- **SET** permet de dire quels sont les champs à modifier et quelles sont les nouvelles valeurs;
- **WHERE** permet de préciser les enregistrements à modifier.

```
UPDATE tblPersonne SET telPers = '418-555-2121'  
WHERE nomPers = 'Gagnon' AND prenPers = 'Jean' ;
```



MODIFICATIONS MULTIPLES

- Il est possible de modifier plusieurs champs par la même commande.

```
UPDATE tblPersonne SET telPers = '418-555-2121' ,  
                                faxPers = '418-555-2020'  
WHERE noPers = 102;
```

- Pour appliquer la modification à tous les enregistrements de la table, il suffit de ne pas mettre de clause **WHERE**.

```
UPDATE tblEnfant SET ageEnfant = ageEnfant +1;
```



SUPPRESSION DE DONNÉES

- La commande permettant de supprimer des données dans une table est **DELETE**.

```
DELETE FROM tblNomTable  
[ WHERE condition ];
```

```
DELETE FROM tblPersonne  
WHERE nomPers = 'Gagnon' AND PrenPers = 'Jean';
```

- Pour vider une table de tous ses éléments, il suffit de ne pas mettre de clause **WHERE**.

```
DELETE FROM tblPersonne;
```



PARTIE2:

DDL (LANGAGE DE DÉFINITION DE DONNÉES)



DÉFINIR LA STRUCTURE D'UNE BD

- Créer une BD
- Créer une table et ses colonnes
 - Les types de données
 - Les contraintes:
 - Colonne obligatoire/facultative
 - Clés primaires
 - Numéro séquentiel généré
 - Clés étrangères
 - Unique
 - Valeurs par défaut
- Supprimer une table
- Supprimer une BD



CRÉER UNE BASE DE DONNÉES

Syntaxe:

```
CREATE DATABASE NomBD ;
```

Exemple:

```
CREATE DATABASE BDPubs ;
```



CRÉER UNE TABLE ET SES COLONNES

tblClient	
noCli:	int
nomCli:	char(30)
adrCli:	char(60)
villeCli:	char(30)
catCli:	char(2)
soldeCli:	num(9,2)

TYPE DE
DONNÉES

```
CREATE TABLE tblClient (noCli      int,
                           nomCli    varchar(30) ,
                           adrCli     varchar(60) ,
                           villeCli   varchar(30) ,
                           catCli     char(2) ,
                           soldeCli   decimal(9,2)) ;
```

LES TYPES DE DONNÉES

- Les principaux types sont:
 - Nombre entier signé ou non (température, quantité commandée, âge)
 - Nombre à virgule (prix, taille)
 - Chaîne de caractères (nom, adresse, article de presse)
 - Date et heure (date de naissance, heure de parution)
 - Énumération (une couleur parmi une liste prédéfinie)
 - Ensemble (une ou des monnaies parmi une liste prédéfinie)



LES TYPES – CHAÎNES

nom	longueur
CHAR(M)	Chaîne de taille fixée à M, où $1 < M < 255$, complétée avec des espaces si nécessaire.
CHAR(M) BINARY	Idem, mais insensible à la casse lors des tris et recherches.
VARCHAR(M)	Chaîne de taille variable, de taille maximum M, où $1 < M < 255$, complété avec des espaces si nécessaire.
VARCHAR(M) BINARY	Idem, mais insensible à la casse lors des tris et recherches.
TINYTEXT	Longueur maximale de 255 caractères.
TEXT	Longueur maximale de 65535 caractères.
MEDIUMTEXT	Longueur maximale de 16777215 caractères.
LONGTEXT	Longueur maximale de 4294967295 caractères.



LES TYPES - ENTIERS

nom	borne inférieure	borne supérieure
TINYINT	-128	127
TINYINT UNSIGNED	0	255
SMALLINT	-32768	32767
SMALLINT UNSIGNED	0	65535
MEDIUMINT	-8388608	8388607
MEDIUMINT UNSIGNED	0	16777215
INT*	-2147483648	2147483647
INT* UNSIGNED	0	4294967295
BIGINT	-9223372036854775808	9223372036854775807
BIGINT UNSIGNED	0	18446744073709551615

(*) : **INTEGER** est un synonyme de **INT**

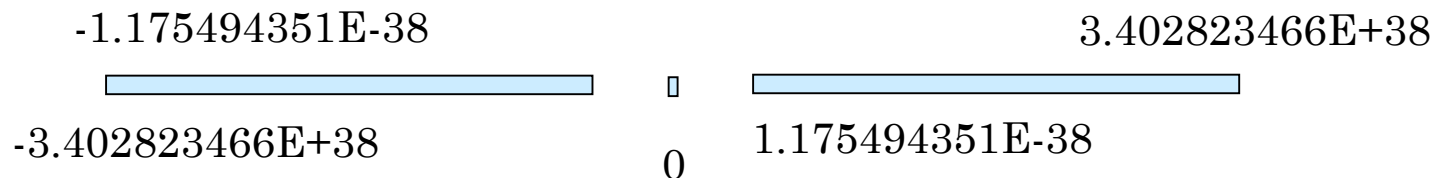
UNSIGNED permet d'avoir un type non signé



LES TYPES – FLOTTANTS

- Les flottants – dits aussi nombres réels – sont des nombres à virgule.

Exemple du type **FLOAT** :



nom	domaine négatif : borne inférieure borne supérieure	Domaine positif : borne inférieure borne supérieure
FLOAT	-3.402823466E+38 -1.175494351E-38	1.175494351E-38 3.402823466E+38
DOUBLE*	-1.7976931348623157E+308 -2.2250738585072014E-308	2.2250738585072014E-308 1.7976931348623157E+308

(*) : **REAL** est un synonyme de **DOUBLE**.

LES TYPES – DATE ET HEURE

nom	description
DATE	Date au format anglophone AAAA-MM-JJ
DATETIME	Date et heure au format anglophone AAAA-MM-JJ HH:MM:SS
TIMESTAMP	Date et l'heure sans séparateur : AAAAMMJJHHMMSS
TIMESTAMP(M)	Idem mais M vaut un entier pair entre 2 et 14. Affiche les M premiers caractères de TIMESTAMP sans compter les 2 premiers de l'année pour M < 12 (voir exemple en bas)
TIME	Heure au format HH:MM:SS
YEAR	Année au format AAAA

Entre **TIMESTAMP(14)** et **TIMESTAMP(12)** ce sont les 2 premiers caractères de l'année qui disparaissent; ensuite on prend les M premiers caractères de **TIMESTAMP(12)**.

TIMESTAMP(14) AAAAMMJJHHMMSS

TIMESTAMP(12) AAMMJJHHMMSS

TIMESTAMP(6) AAMMJJ

TIMESTAMP(4) AAMM

TIMESTAMP(2) AA



LES CONTRAINTES – OBLIGATOIRE/FACULTATIVE

- Une colonne est **facultative** par défaut.
- Il faut donc déclarer explicitement les colonnes **obligatoires**.

```
CREATE TABLE tblClient (NoCli      int      NOT NULL,  
                          NomCli    varchar(30) NOT NULL,  
                          AdrCli    varchar(60) NOT NULL,  
                          VilleCli  varchar(30) NOT NULL,  
                          CatCli    char(2)   NULL,  
                          SoldeCli  decimal(9,2) NOT NULL  
);
```



Facultatif



LES CONTRAINTES - CLÉ PRIMAIRE (PK)

Première méthode

tblClient	
<u>NoCli</u> :	int
NomCli:	char(30)
AdrCli:	char(60)
VilleCli:	char(30)
CatCli:	char(2)
SoldeCli:	num(9,2)

```
CREATE TABLE tblClient (NoCli      int           NOT NULL,
                          NomCli    varchar(30)   NOT NULL,
                          AdrCli     varchar(60)   NOT NULL,
                          VilleCli   varchar(30)   NOT NULL,
                          CatCli     char(2)       NULL,
                          SoldeCli   decimal(9,2) NOT NULL
                          PRIMARY KEY (NoCli)
                          );
```

LES CONTRAINTES - CLÉ PRIMAIRE (PK)

Deuxième méthode

tblArticle	
<u>NoArt</u> :	int
TitreArt:	char(80)
TxtArt:	texte
DateArt:	date
AutArt:	char(80)

```
CREATE TABLE tblArticle
( NoArt medium int unsigned PRIMARY KEY,
  TitreArt varchar(80),
  TxtArt text,
  DateArt date,
  AutArt varchar(80)
);
```

NUMÉRO SÉQUENTIEL GÉNÉRÉ

```
CREATE TABLE tblPersonne
(
    NoPers smallint unsigned PRIMARY KEY AUTO_INCREMENT,
    NomPers    varchar(40) ,
    PrenPers   varchar(40) ,
    AdrPers    tinytext,
    TelPers    char(12)
);
```

- L'utilisation de AUTO_INCREMENT permet de générer automatiquement un numéro séquentiel pour la « *clé primaire* ».
- La numérotation débute à 1 et le pas est de 1.

PERSONNE

NoPers	NomPers	PrenPers	AdrPers	TelPers
1	Dupond	Marc	8 rue du Pont	418-544-5454
2	Dupont	Pierre	14 boul Dupond	418-555-4444




LES CONTRAINTES - CLÉ PRIMAIRE COMBINÉE

- La clé primaire peut être composée de plusieurs champs


Mauvaise syntaxe :

```
CREATE TABLE tblPersonne
  (NomPers      varchar(40) PRIMARY KEY,
   PrenPers     varchar(40) PRIMARY KEY,
   TelPers      char(10));
```

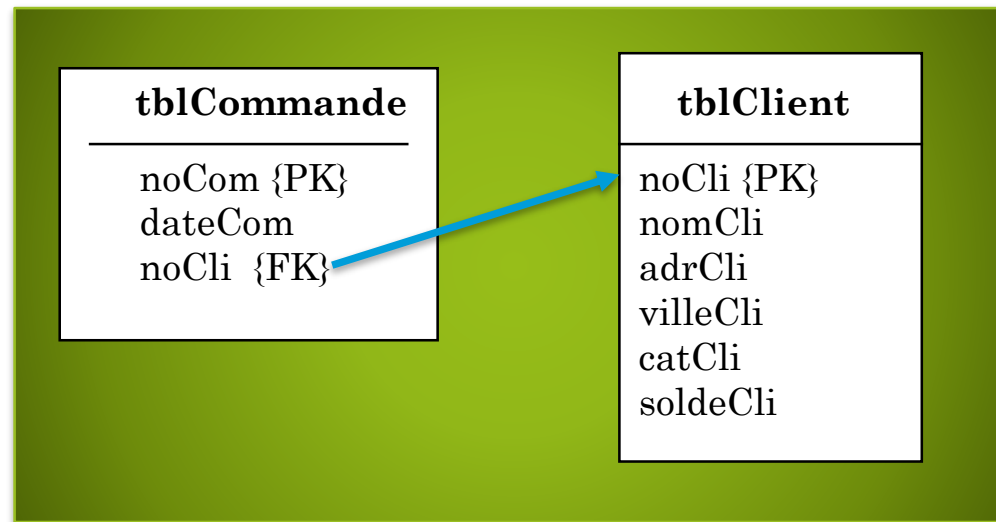


Bonne syntaxe :

```
CREATE TABLE tblPersonne
  (NomPers      varchar(40),
   PrenPers     varchar(40),
   TelPers      char(10),
   PRIMARY KEY (NomPers, PrenPers));
```



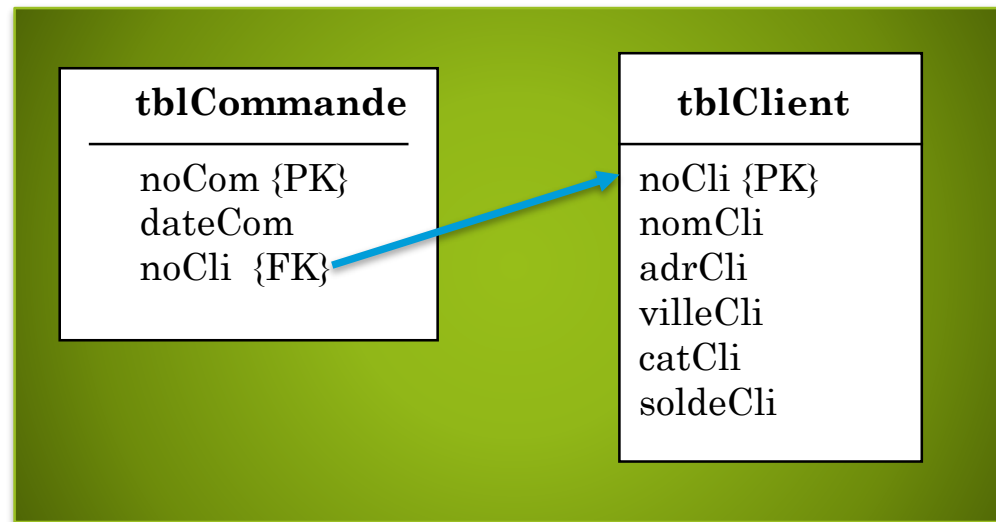
LES CONTRAINTES - CLÉ ÉTRANGÈRE (FK)



```
CREATE TABLE tblCommande
(
    noCom      char(12) NOT NULL,
    noCli      char(10) NOT NULL,
    dateCom    date     NOT NULL,
    PRIMARY KEY (noCom),
    FOREIGN KEY (noCli) REFERENCES tblClient(noCli);

```

LES CONTRAINTES - CLÉ ÉTRANGÈRE (FK)



```
CREATE TABLE tblCommande
  (noCom      char(12) PRIMARY KEY,
   noCli      char(10) NOT NULL REFERENCES tblClient(noCli),
   dateCom    date      NOT NULL
  );
```

LES CONTRAINTES - UNIQUE

- Pour interdire l'apparition de doublons pour un champ, on utilise l'option UNIQUE.

UNIQUE (*liste des champs*)

UNIQUE (NomPers)

- Pour interdire les doublons sur 2 champs mais en les laissant indépendants :

UNIQUE(NomPers)

UNIQUE(PrenPers)

Enregistrement interdit car
'Marc' est un doublon dans
la colonne 'PrenPers'

NomPers	PrenPers
Dupond	Marc
Dupont	Pierre
Martin	Marc

LES CONTRAINTES – UNIQUE COMBINÉ

- Pour interdire les doublons d'une combinaison de champs, on liste l'ensemble des champs dans une seule commande **UNIQUE**.

Exemple: pour interdire tout doublon du couple *nom, prénom*:

UNIQUE(NomPers,PrenPers)

NomPers	PrenPers
Dupond	Marc
Dupont	Pierre
Martin	Marc
Martin	Pierre
Martin	Marc

Enregistrement interdit car le couple ('Martin', 'Marc') existe déjà



CONTRAINTE D'UNICITÉ

- Pour qu'un champs soit clé secondaire, on utilise une contrainte d'unicité pour s'assurer que les valeurs de la colonne soient uniques au sein de la table.
- Ce champ peut accepter ou non les valeurs nulles.

```
CREATE TABLE tblClient (noCli      char(10)      NOT NULL,  
                          nomCli     varchar(30)   NOT NULL,  
                          adrCli     varchar(60)   NOT NULL,  
                          villeCli   varchar(30)   NULL,  
                          catCli     char(2),  
                          soldeCli   decimal(9,2) NOT NULL,  
                          PRIMARY KEY (noCli),  
                          UNIQUE (nomCli)  
);
```

UNE AUTRE FAÇON DE L'ÉCRIRE

- Comme pour les clés, il est possible d'inscrire la contrainte suite à la déclaration de la colonne.

```
CREATE TABLE tblClient (noCli      char(10)      NOT NULL,  
                          nomCli     varchar(30)    NOT NULL UNIQUE,  
                          adrCli     varchar(60)    NOT NULL,  
                          villeCli  varchar(30)    NULL,  
                          catCli     char(2),  
                          soldeCli  decimal(9,2) NOT NULL),  
                          PRIMARY KEY (noCli)  
  
);
```



CONTRAINTE VALEUR PAR DÉFAUT

- La contrainte DEFAULT permet de déterminer la valeur qui sera assignée à la colonne si on ne spécifie pas de valeur;

```
CREATE TABLE tblClient
    (noClient      char(10)      NOT NULL,
     nomClient     varchar(30)   NOT NULL,
     adrClient     varchar(60)   NOT NULL,
     villeClient   varchar(30)   NOT NULL DEFAULT 'Paris',
     catClient     char(2)       DEFAULT 'B1',
     soldeClient   decimal(9,2)  NOT NULL DEFAULT 0.0
    );
```



SUPPRIMER UNE TABLE

Syntaxe:

```
DROP TABLE NomTable ;
```

Exemples:

```
DROP TABLE tblCommande ;
```

```
DROP TABLE tblPersonne;
```

Attention, opération sous haute surveillance !
La table ne doit plus être référencée par une clé étrangère



SUPPRIMER UNE BASE DE DONNÉES

Syntaxe:

```
DROP DATABASE NomBD ;
```

Exemples:

```
DROP DATABASE BDPubs ;
```

```
DROP DATABASE BDInventaire ;
```

Attention ! Tout sera perdu!
S'assurer d'avoir un script de création

