TP2 : Construire des tables…. **Les Types de données(champs)**

1 Objectif

- Objectif : Présentation et utilisation des **types de données : entier, réel, chaîne de caractères, date**.
- Vous pouvez vous aider :
  - de cette documentation sur w3school ou de cette documentation sur openclassroom
  - de la documentation de mysql , et la documentation de mariadb , et la documentation de SQLserver , ou encore la documentation de Oracle ….
  - de ce petit résumé sur cours-info

Rappel : Depuis le terminal, se connecter au serveur **mysql**

```
mysql --user=votreLogin --password=JJMM --host=serveurmysql --database=BDD_votreLogin
```

- Créer un fichier et **prendre des notes** pour les contrôles

2 Présentation des différents types de données dans MySQL

Les types standards (fondamentaux) sont :

- **INT** ou(alias) **INTEGER**, SMALLINT, BIGINT, ….
- **DECIMAL(X,Y)** ou(alias) **NUMERIC(X,Y)**
- FLOAT(X) ou(alias) REAL
- CHAR(X)
- **VARCHAR(X)**
- **DATE** (AAAA-MM-JJ)
- DATETIME (AAAA-MM-JJ HH:MM:SS)

2.1 les entiers

type "Numérique"

Access	MySQL	Oracle	PostgreSQL	SQLserver
INT	INT	NUMBER(10)	INTEGER	INT
byte	tinyint	NUMBER(3)	tinyint	tinyint
smallint	smallint	NUMBER(5)	smallint	smallint
BIGINT (large number)	BIGINT	NUMBER(19)	BIGINT	BIGINT

- INT (alias INTEGER) : entier codé sur 4 octets (-2 147 438 648 à 2 147 438 647)
- BYTE : entier codé sur 1 octet (0 à 255)
- SMALLINT : entier codé sur 2 octets (-32 768 à 32767)

- Test sur des numériques de type *Entier* : Recopier le script ci dessous sur un terminal connecté à MySQL, répondre aux questions ci dessous à l'aide de la documentation.
- Dans un enregistrement, si un champ est de type **entier (INT)** :
  - Quel est le contenu du champ si on enregistre un entier plus grand que la capacité de stockage de ce champ ?
  - À quoi sert le chiffre entre parenthèses derrière le type, a-t-il toujours une utilité ?
  - Quel est le rôle de l'attribut **UNSIGNED** ?

```
CREATE TABLE test_int (
x1 INT,
x2 INT(6)
);
INSERT INTO test_int (x1,x2) VALUES (1,1);
INSERT INTO test_int (x1,x2) VALUES (12,12);
INSERT INTO test_int (x1,x2) VALUES (12345678,12345678);
INSERT INTO test_int (x1,x2) VALUES (123456789,-123456789);
INSERT INTO test_int (x1,x2) VALUES (1234567890,-1234567890);
INSERT INTO test_int (x1,x2) VALUES (9999999999,9999999999);
```

```
SELECT x1,x2 FROM test_int;
DESCRIBE test_int;
DROP TABLE test_int;

# remplacer
# par
# x INT
# puis par
# x TINYINT     => conclusion :
```

les types “entiers”

- TINYINT : 1 octet – valeurs : de -128 à 127 – valeur (entier non signé) 0 à 255
- SMALLINT : 2 octets – valeurs : de -32 768 à 32 767 – valeur (entier non signé) 0 à 65 535
- INT ou INTEGER : 4 octets – valeurs : de -2 147 483 648 à 2 147 483 647 – valeur (entier non signé) 0 à 4 294 967 295
- BIGINT : 8 octets – valeurs : de 9 223 372 036 854 775 808 à 9 223 372 036 854 775 807

Pour les champs de type numérique, l’option **UNSIGNED** permet de stocker des nombres 2 fois plus grands mais positifs (on ne stocke plus le bit de signe).
ZEROFILL : définit le caractère de remplissage par défaut, cette option définit l’entier en non signé.
Pour la déclaration avec l’option ZEROFILL : TYPE(M) [unsigned] avec M : nombre de décimales signifiantes qui seront affichées.
exemple : **telephone INT(10) ZEROFILL**, cependant **il vaut mieux enregistrer un numéro de téléphone dans une chaîne de caractères**.

2.2 les chiffres à virgules

type “Numérique”

Access	MySQL	Oracle	PostgreSQL	SQL server
decimal(n,d)	decimal(n,d)	**number(n,d)**	numeric(n,d)	decimal(n,d)

- decimal(n,d) : nombre décimal de “n” chiffres dont “d” après la virgule, un alias est **numeric**

- Cas de **NUMERIC ou(alias) DECIMAL**, la valeur stockée est la valeur numérique exacte (ce n’est pas le cas avec un réel). Attention aux limites liées à la déclaration. Pour plus d’informations, documentation MySQL

Conseil : **FLOAT versus DECIMAL** Il est conseillé d’utiliser DECIMAL qui est un nombre exact, plutôt que FLOAT qui est un nombre approximatif, si la précision requise est suffisante. FLOAT sera réservé typiquement à des calculs scientifiques nécessitant un degré de précision supérieur.

- Test sur des réels, répondre aux questions ci dessous à l’aide de la documentation en ligne.
- Quel est le contenu du champ si on enregistre une valeur plus grand ou plus petit que la capacité de stockage du champ ?
- À quoi servent les chiffres entre parenthèses derrière le type ?

```
CREATE TABLE test_numeric (
y NUMERIC(4,2)
);

INSERT INTO test_numeric (y) VALUES (12.2222);
INSERT INTO test_numeric (y) VALUES (-12.222);
INSERT INTO test_numeric (y) VALUES (123.2222);
INSERT INTO test_numeric (y) VALUES (999.2222);
INSERT INTO test_numeric (y) VALUES (999.9944);
INSERT INTO test_numeric (y) VALUES (999.9955);
INSERT INTO test_numeric (y) VALUES (9999999999.2222);
INSERT INTO test_numeric (y) VALUES (-9999999999.2222);

SELECT * FROM test_numeric;
```

```
DESCRIBE test_numeric;
DROP TABLE test_numeric;

# y NUMERIC(6,2) ZEROFILL    => conclusion :

# Important : différence entre NUMERIC et FLOAT :
```

type "Numérique" : réel

Access	MySQL	Oracle	PostgreSQL	SQL server
real	FLOAT	FLOAT(49) ou FLOAT(23)	real	FLOAT
double	double	binary_double	double precision	real

- float (alias real) : nombre réel codé sur 4 octets (1 bit de signe, 8 bits pour l'exposant et 23 bits pour la mantisse)
- double : nombre réel codé sur 8 octets (1 bit de signe, 11 bits pour l'exposant et 52 bits pour la mantisse)

REMARQUE : différence entre FLOAT et REAL sur 'SQL server'

```
CREATE TABLE test_float (
y FLOAT(4,2)
);

INSERT INTO test_float (y) VALUES (12.2222);
INSERT INTO test_float (y) VALUES (-12.222);
INSERT INTO test_float (y) VALUES (123.2222);
INSERT INTO test_float (y) VALUES (999.2222);
INSERT INTO test_float (y) VALUES (999.9944);
INSERT INTO test_float (y) VALUES (999.9955);
INSERT INTO test_float (y) VALUES (9999999999.2222);
INSERT INTO test_float (y) VALUES (-9999999999.2222);

SELECT * FROM test_float;
DESCRIBE test_float;
DROP TABLE test_float;
```

NUMERIC (enregistre la valeur exacte d'un réel) DECIMAL[(M[,D])] est équivalent à NUMERIC [(M,D)] Occupe M+2 octets si D > 0, M+1 octets si D = 0
Contient des nombres flottants stockés comme des chaînes de caractères.

REEL : FLOAT, DOUBLE, REAL: Stocke un nombre de type flottant.(-1.175494351E-38 à 3.402823466E+38), avec Signe Exposant Mantisse (FLOAT : 4 octets DOUBLE : 8 octets) Exemple float(4,2)=> valeur maximum 99,99

2.3 Les chaînes de caractères

type "TEXTE"

Access	MySQL	Oracle	PostgreSQL
varchar(n)	varchar(n)	varchar2(n)	varchar(n)
char(n)	char(n)	char(n)	char(n)
text	text	text	clob

- varchar(n) : Chaine alphanumérique à longueur variable de 1 à 255 caractères (n taille maxi)
- char(n) : Chaine alphanumérique à longueur fixe de 1 à 255 caractères (complèter par des blancs si la taille est inférieur à la taille maximale n)
- text : zone de texte jusqu'à 65536 caractères

- Test sur des chaînes de caractères, répondre aux questions ci dessous à l'aide de la documentation en ligne.
- Dans un enregistrement, si un champ est de type **chaîne de caractères (CHAR VARCHAR TEXT …)** :
  - Quelle est la différence entre **CHAR** et **VARCHAR** ?
  - Quelle est le nombre de caractères maximum que peut enregistrer un type TEXT ?

- **ENUM** n'est pas standard à tous les SGBDR, mais consulter la documentation de MYSQL ou les informations sur le site d'openclassroom

```
CREATE TABLE test_char (
c VARCHAR(7)
);

INSERT INTO test_char (c) VALUES ('essai essai');
INSERT INTO test_char (c) VALUES ('Monsieur Monsieur Monsieur        ' );
INSERT INTO test_char (c) VALUES ('M.' );

SELECT * FROM test_char;
DESCRIBE test_char;
DROP TABLE test_char;

# remplacer par le type ci dessous le type de l'enregistrement
# c CHAR(13)               => conclusion :
# c TEXT
```

2.4 les dates/heures

type "date/heure"

Access	MySQL	Oracle	PostgreSQL
date	date	date	date
time	time	date	time
datetime	datetime	date	timestamp

- date : Date codée sur 4 octets
- time : Heure codée sur 8 octets
- datetime : Date et heure codée sur 12 octets

- Test sur des dates, répondre aux questions ci dessous à l'aide de la documentation en ligne.
- Dans un enregistrement, si un champ est de type **date (DATE DATETIME TIMESTAMP TIME)** :
  - Quelle est la valeur maximum et minimum dans un champ de type TIME ?
  - Donner les **formats** pour enregistrer un champ dans un enregistrement sur **DATE**, **TIME** et **DATETIME**
  - Quelle est la différence entre les types **DATETIME** et **TIMESTAMP** ? (voir bug de l'an 2038)
  - Quel est l'intérêt de mettre la date courante lors de la création de l'enregistrement ? (voir remarque ci dessous)
  - Quelle est la date mini et maxi pour **TIMESTAMP**, consulter la documentation : http://dev.mysql.com/doc/refman/5.7/en/date-and-time-types.html
- **Prendre des notes** pour les contrôles

```
CREATE TABLE test_date1 (
date1 DATE
);
INSERT INTO test_date1 (date1) VALUES ('2019-03-1');
INSERT INTO test_date1 (date1) VALUES ('2019-13-13');
INSERT INTO test_date1 (date1) VALUES ('01-03-2019');
INSERT INTO test_date1 (date1) VALUES ('2017/03/19');
INSERT INTO test_date1 (date1) VALUES ('2019#03#25');
INSERT INTO test_date1 (date1) VALUES ('3200-03-26');
INSERT INTO test_date1 (date1) VALUES ('200-03-27');
INSERT INTO test_date1 (date1) VALUES ('0000-00-00');
INSERT INTO test_date1 (date1) VALUES ('-200-03-28');
SELECT * FROM test_date1;
DESCRIBE test_date1;
DROP TABLE test_date1;


CREATE TABLE test_heure1 (
heure1 TIME
);
INSERT INTO test_heure1 (heure1) VALUES ('14:30');
INSERT INTO test_heure1 (heure1) VALUES ('14:30:55');
INSERT INTO test_heure1 (heure1) VALUES ('24:00');
INSERT INTO test_heure1 (heure1) VALUES ('28:00');
```

```
INSERT INTO test_heure1 (heure1) VALUES ('700:59:59');
INSERT INTO test_heure1 (heure1) VALUES ('99:60:59');
INSERT INTO test_heure1 (heure1) VALUES ('838:59:59'); -- 839
SELECT * FROM test_heure1;
DESCRIBE test_heure1;
DROP TABLE test_heure1;


CREATE TABLE test_date2 (
date2 TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
INSERT INTO test_date2 (date2) VALUES ('2019-03-1');
INSERT INTO test_date2 (date2) VALUES (NULL);
INSERT INTO test_date2 (date2) VALUES ('2017-10-10 14:30:55');
INSERT INTO test_date2 (date2) VALUES ('2019-13-13');
INSERT INTO test_date2 (date2) VALUES ('01-03-2019');
INSERT INTO test_date2 (date2) VALUES ('2017/03/19');
INSERT INTO test_date2 (date2) VALUES ('2019#03#25');
INSERT INTO test_date2 (date2) VALUES ('3200-03-26');
INSERT INTO test_date2 (date2) VALUES ('200-03-27');
INSERT INTO test_date2 (date2) VALUES ('0000-00-00');
INSERT INTO test_date2 (date2) VALUES ('-200-03-28');
SELECT * FROM test_date2;
DESCRIBE test_date2;
DROP TABLE test_date2;



CREATE TABLE test_date3 (
date3 DATETIME
);
INSERT INTO test_date3 (date3) VALUES ('2019-03-1');
INSERT INTO test_date3 (date3) VALUES (NULL);
INSERT INTO test_date3 (date3) VALUES ('2017-10-10 14:30:55');
INSERT INTO test_date3 (date3) VALUES ('2019-13-13');
INSERT INTO test_date3 (date3) VALUES ('01-03-2019');
INSERT INTO test_date3 (date3) VALUES ('2017/03/19');
INSERT INTO test_date3 (date3) VALUES ('2019#03#25');
INSERT INTO test_date3 (date3) VALUES ('3200-03-26');
INSERT INTO test_date3 (date3) VALUES ('200-03-27');
INSERT INTO test_date3 (date3) VALUES ('0000-00-00');
INSERT INTO test_date3 (date3) VALUES ('-200-03-28');
SELECT * FROM test_date3;
DESCRIBE test_date3;
DROP TABLE test_date3;

## conclusion

# format DATE  =>
# format TIME =>
# format TIMESTAMP =>
# format DATETIME =>
```

- Remarque : **les valeurs par défaut sont des constantes**, sauf une exception uniquement sur le SGBDR MySql : pour un champ de type **TIMESTAMP** la valeur peut être par défaut **CURRENT_TIMESTAMP**
- Remarque 2 : attention aux champs de type *DATETIME* : si vous recherchez des enregistrements compris entre 2 dates (exemple des vols), vous ne prenez pas en compte le dernier jour.

2.5 les booleens

valeur “vrai” ou “faux” codée sur un octet

Access	MySQL	Oracle	PostgreSQL	SQL server
logical	tinyint(1)	number(1)	booleen	byte

- sur mysql, c’est un entier dans lequel on place “0” ou “1”
- sur mariadb, il existe des alias, mais ces alias ne fonctionnent pas obligatoirement sur mysql d’Oracle

```
CREATE TABLE test_bool (
x1 bool,
x2 boolean
);
INSERT INTO test_bool (x1,x2) VALUES (true,false);
INSERT INTO test_bool (x1,x2) VALUES (1,-1);
INSERT INTO test_bool (x1,x2) VALUES (12,-12);
INSERT INTO test_bool (x1,x2) VALUES (123,-123);

SELECT x1,x2 FROM test_bool;
DESCRIBE test_bool;
DROP TABLE test_bool;
```

2.6 les autres types

currency : valeur monétaire codée sur 8 octets comprenant 15 chiffres + 4 décimales

Access	MySQL	Oracle	PostgreSQL	SQL server
currency	decimal(19,4)	number(19,4)	money	money