1.NSI **Représentation des nombres entiers**

Exercice 1: Déterminer l'écriture en base 2 des entiers a, b et c.
En déduire le nombre de bits nécessaire au stockage de ces entiers.

$a = 31$

$b = 32$

$c = 2^{10}$

Exercice 2 :
a) Déterminer l'écriture décimale du plus grand entier naturel dont l'écriture binaire comporte 6 chiffres.
b) Déterminer de même l'écriture décimale du plus grand entier naturel qu'on peut stocker sur 16 bits.

Intervalles d'entiers naturels sur *n* bits
Compléter le tableau :

nombre d'octets	nombre de bits	entiers naturels :	nombre d'entiers stockés
1	8	de 0 à 255	256
	16		
4			
	64		

Exercice 3 :
Soit x un entier naturel stocké sur 7 bits. Soit y un entier naturel stocké sur 5 bits.
a) Evaluer le nombre de bits nécessaires au stockage de $x+y$
b) Evaluer le nombre de bits nécessaires au stockage de $x \times y$

Exercice 4 :
Soit x un entier stocké sur 16 bits. Soit y un entier stocké sur 32 bits.
a) Evaluer le nombre de bits nécessaires au stockage de $x+y$
b) Evaleur le nombre de bits nécessaires au stockage de $x \times y$

Activité : Encodage des entiers négatifs
Vocabulaire : un *mot binaire* de longueur *n* est une suite de *n* chiffres binaires (ou « bits »).
En base 2, un mot binaire est associé à un entier naturel, qu'on notera bin(m).

Pour stocker en mémoire un entier négatif, il faut décider quel mot binaire on lui associe.
Pour les entier positifs, cela ne pose aucun problème.
Mais pour représenter des entiers négatifs, plusieurs choix sont possibles.

Le tableau suivant présente trois méthodes pour représenter des entiers relatifs sur 4 bits.

Entier naturels		**Entiers relatifs**					
4 bits	**Binaire non signé**	**4 bits**	**Décalage -7**	**4 bits**	**Binaire signé**	**4 bits**	**Binaire complément à 2**
0 0 0 0	0	0 0 0 0	-7	0 0 0 0	0	0 0 0 0	0
0 0 0 1	1	0 0 0 1	-6	0 0 0 1	1	0 0 0 1	1
0 0 1 0	2	0 0 1 0	-5	0 0 1 0	2	0 0 1 0	2
0 0 1 1	3	0 0 1 1	-4	0 0 1 1	3	0 0 1 1	3
0 1 0 0	4	0 1 0 0	-3	0 1 0 0	4	0 1 0 0	4
0 1 0 1	5	0 1 0 1	-2	0 1 0 1	5	0 1 0 1	5
0 1 1 0	6	0 1 1 0	-1	0 1 1 0	6	0 1 1 0	6
0 1 1 1	7	0 1 1 1	0	0 1 1 1	7	0 1 1 1	7
1 0 0 0	8	1 0 0 0	1	1 0 0 0	-0	1 0 0 0	-8
1 0 0 1	9	1 0 0 1	2	1 0 0 1	-1	1 0 0 1	-7
1 0 1 0	10	1 0 1 0	3	1 0 1 0	-2	1 0 1 0	-6
1 0 1 1	11	1 0 1 1	4	1 0 1 1	-3	1 0 1 1	-5
1 1 0 0	12	1 1 0 0	5	1 1 0 0	-4	1 1 0 0	-4
1 1 0 1	13	1 1 0 1	6	1 1 0 1	-5	1 1 0 1	-3
1 1 1 0	14	1 1 1 0	7	1 1 1 0	-6	1 1 1 0	-2
1 1 1 1	15	1 1 1 1	8	1 1 1 1	-7	1 1 1 1	-1

Pour chaque représentation, identifier les avantages et inconvénients.

a) Représentation par décalage

Une valeur d étant fixée (« décalage »), la valeur associée à un mot binaire m est : bin(m) – d

Dans l'exemple : d = 7 donc le mot binaire `0 1 1 1` représente 7 – 7 soit 0

Avantage : …

Inconvénients :

b) Représentation binaire signée

Le premier bit est réservé au signe, les 3 autres bits à la « valeur absolue ».

Avantage : …

Inconvénients :

-
-

c) Représentation « complément à 2 »

Chaque bit représente une puissance de 2, mais le bit de poids fort, représente -8 (et non +8)

poids	**–8**	**4**	**2**	**1**	**calcul**
chiffres binaires	0	1	1	0	

poids	**–8**	**4**	**2**	**1**	**calcul**
chiffres binaires	1	1	1	0	

Inconvénient : …

Avantages

- l'addition (-1) + 1 donne ...
- au niveau du processeur,

Mot binaire sur **4 bits**	Entier relatif
1 1 1 1	-1
+ 0 0 0 1	1
(1)0 0 0 0	0

Exercice 5 : dimensionner des variables de type entier (relatif) dans une base de données
Pour stocker des informations dans une base de données, Lisa doit définir le type de chaque donnée. Sur le site openclassrooms, elle trouve les informations suivantes (certaines valeurs du tableau ont été effacées)

Nombres entiers

Les types de données qui acceptent des nombres entiers comme valeurs sont désignés par le mot-clé `INT`, et ses déclinaisons `TINYINT`, `SMALLINT`, `MEDIUMINT` et `BIGINT`. La différence entre ces types est le nombre d'octets (donc la place en mémoire) réservés à la valeur du champ. Voici un tableau reprenant ces informations, ainsi que l'intervalle dans lequel la valeur peut être comprise pour chaque type.

Type	Nombre d'octets	Minimum	Maximum
`TINYINT`	1	-128	127
`SMALLINT`	2	-32768	
`MEDIUMINT`	3		
`INT`	4	-2147483648	2147483647
`BIGINT`	8	-9223372036854775808	9223372036854775807

Si vous essayez de stocker une valeur en dehors de l'intervalle permis par le type de votre champ, MySQL stockera la valeur la plus proche. Par exemple, si vous essayez de stocker 12457 dans un `TINYINT`, la valeur stockée sera 127 ; ce qui n'est pas exactement pareil, vous en conviendrez. Réfléchissez donc bien aux types de vos champs.

L'attribut `UNSIGNED`

Vous pouvez également préciser que vos colonnes sont `UNSIGNED`, c'est-à-dire que l'on ne précise pas s'il s'agit d'une valeur positive ou négative (on aura donc toujours une valeur positive). Dans ce cas, la longueur de l'intervalle reste la même, mais les valeurs possibles sont décalées, le minimum valant 0. Pour les `TINYINT`, on pourra par exemple aller de 0 à 255.

Source : https://openclassrooms.com/fr/courses/1959476-administrez-vos-bases-de-donnees-avec-mysql/1960456-distinguez-les-differents-types-de-donnees Consulté le 8 avril 2020 à 13h41

1) Compléter le tableau ci-dessus en calculant les valeurs manquantes.
2) Déterminer la plage de valeurs correspondant à une donnée de type :
   a) SMALLINT UNSIGNED
   b) MEDIUMINT UNSIGNED
3) On suppose que A et B sont des données de type TINYINT. Peut-on toujours stocker dans une donnée de type TINYINT le résultat du calcul A+B ? Justifier
4) Déterminer quel type il faut définir pour la donnée C, si on souhaite y stocker sans erreur le résultat du calcul A x B dans les cas suivants :
   a) A est B sont de type TINYINT
   b) A est de type TINYINT UNSIGNED, et B de type SMALLINT UNSIGNED
   c) A est de type TINYINT, et B de type SMALLINT.