Exercice sur la création des bases de données et y insérer des données

|  |  |
| --- | --- |
| Compétences | Insérer (INSERT INTO) des données dans des bases de données via le langage SQL |
| Objectifs | A la fin des exercices, l’élève doit être capable :   * D’introduire des données dans les TABLES de la base de données. * De comprendre et d’appliquer correctement le séquencement prioritaire des ordres d’insertion. * De modifier des données. * De supprimer des données. * D’appliquer la suppression en cascade sur les différentes clés étrangères. |
| Durée | 2 périodes de 45 min avec la correction |
| Répertoire de travail | **Devoirs Teams ou SharePoint ETML-INF\Supports de cours\ICT\164-CreerBD-InsertionDonnees\b-UnitesEnseignement\OO4-InsertionDonnées** |
| Fichiers sources dans le répertoire de travail | * Un fichier E-164-Ex2\_1\_insertion\_donnees\_cols.docx avec les présentes informations * Autres fichiers expliqués dans ce document |

[Contexte 2](#_Toc126788453)

[1 Demande 3](#_Toc126788454)

[1.1 Créer la base de données initiale à l’aide du fichier 4](#_Toc126788455)

[1.2 Pour insérer des données 4](#_Toc126788456)

[1.3 « Casser » la feuille totale EXCEL en sous-feuilles 4](#_Toc126788457)

[1.4 Enlever les doublons dans les données EXCEL 5](#_Toc126788458)

[1.5 Calculer les cellules de manière à obtenir le code SQL d’insertion 6](#_Toc126788459)

[1.6 Constituer votre fichier d’insertion des données 7](#_Toc126788460)

[1.7 Contraintes sur les scripts de création (fourni) et d’insertion 8](#_Toc126788461)

[1.8 Création des références et de l’intégrité référentielle 8](#_Toc126788462)

[1.9 Suppression des champs de la TABLE « tbl\_Cols » ayant servi à créer les références 9](#_Toc126788463)

[1.10 CREATION DES CLES ETRANGERES et de l'intégrité référentielle 9](#_Toc126788464)

[2 Contrôle des DONNEES 9](#_Toc126788465)

[2.1 Contrôle des données de chaque TABLE 9](#_Toc126788466)

[2.2 Contrôle des données introduites par une requête 10](#_Toc126788467)

# Contexte

Nous sommes en 2012 : Vous êtes passionné de vélo et vous en train de gravir le col du Sanetch avec un copain, et discuté tout en pédalant.

Soudain dans la discussion, il vient le propos qu’il existe un « club des 100 cols » et l’idée de pour voir faire partie des leurs vous traverse l’esprit…

Pour cela faut-il encore réaliser 100 cols reconnus comme tel ou plus encore …

Une fois de retour au bercail et en surfant sur la toile vous tombez sur le site Wikipedia qui répertorie les cols routiers, et pour la Suisse uniquement, il y en a environ 107, cool… !

Le « challenge » de faire tous les cols routiers de Suisse prend forme et vous êtes prêt à le relever en vous basant sur ceux répertoriés par [Wikipédia](https://fr.wikipedia.org/wiki/Liste_des_cols_de_Suisse).

Vous vous fixé 5 ans pour réaliser ce « challenge ».

Il vous faut donc une base de données qui vous permettent de tenir à jour votre « compteur ».

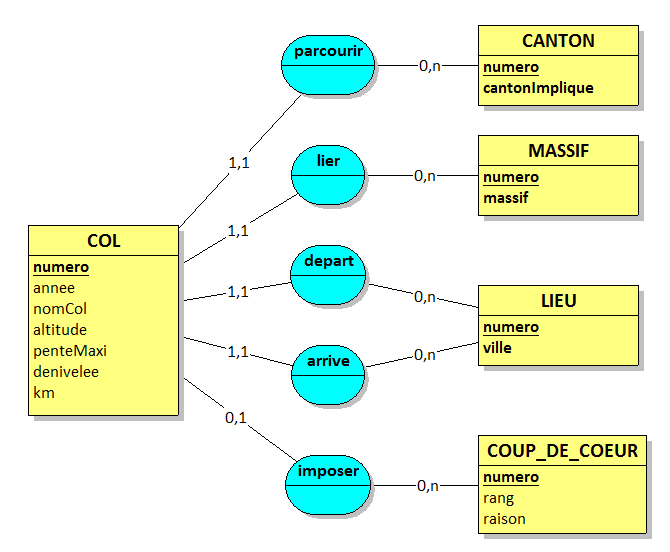
C’est là que vous intervenez.

Une fois la base de données mise en place, il vous faudra introduire les cols que vous avez déjà fait jusqu’à présent.

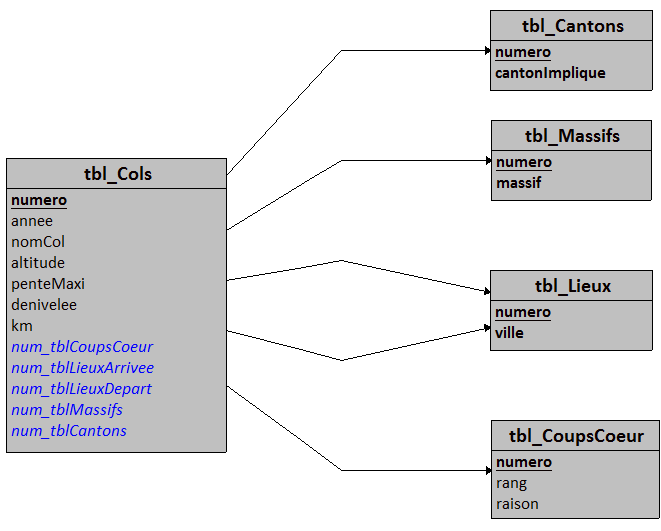
1. La base de données vous sera transmise sous la forme d’un fichier de configuration.
2. Les données à introduire sous la forme d’un fichier excel.

# Demande

MCD imposé



MLD



## Créer la base de données initiale à l’aide du fichier

Ayant le fichier « .sql » avec les ordres de création vous arrivez facilement à créer les TABLES. Faites-le en premier.

Dans un 1er temps, nous ferons la TABLE « tbl\_Cols » sans appliquer l’intégrité référentielle et ses FK. La TABLE « tbl\_Cols » gardera les champs « cantonImplique, massif, villeDepart, villeArrivee, rang » pour pouvoir ensuite créer les clés étrangères (FK).

Pas besoin du champ « raison », on travaillera sur le « rang ».

Dans un 2ème temps, nous modifierons cette TABLE.

## Pour insérer des données

* On vous a transmis un fichier EXCEL avec 22 cols à introduire.
* Dans un 1er temps, il vous faut trouver un moyen pour fragmenter les données « dites à plat » du fichier EXCEL pour les insérer dans la TABLE « tbl\_Cols » et les TABLES « périphériques ».
* Il faudra aussi travailler avec les PK et FK de manière à garantir l’intégrité référentielle.
* C’est là qu’il faut commencer à réfléchir sur la stratégie à appliquer…

Ce qu’on sait déjà…

L’ordre SQL pour insérer des données dans une TABLE est : INSERT INTO…

Il peut s’appliquer de la sorte :

**INSERT INTO nom de la table (colonne1, colonne2,…) VALUES(valeurs1, valeur2,…)**

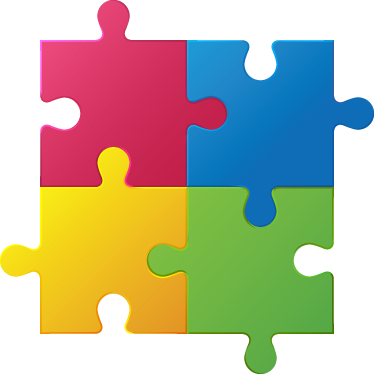
On peut donc à l’aide d’EXCEL faire un « calcul de cellules » de manière à composer ces ordres dans une autre cellule EXCEL et ceci ligne par ligne.

Ensuite copiant ces ordres dans un fichier « .sql » les exécuter dans MySQL.

C’est la stratégie que nous allons utiliser…

## « Casser » la feuille totale EXCEL en sous-feuilles

La totalité de la feuille doit être répartie en plusieurs autres feuilles n’ayant pas de doublon et qui correspondront aux TABLES périphériques :

* tbl\_Cols
* tbl\_Cantons
* tbl\_Massifs
* tbl\_Lieux
* tbl\_CoupsCoeurs

Une fois que vous avez les 5 feuilles copiées avec les mêmes données, ne garder que les colonnes liées aux données désirées en renommant la « feuille » EXCEL avec le nom de la TABLE-future (tbl\_Cantons, tbl\_Massifs, tbl\_Lieux, tbl\_CoupsCoeur, tbl\_Cols).

## Enlever les doublons dans les données EXCEL

Les données de chaque feuille comportent des « doublons ».

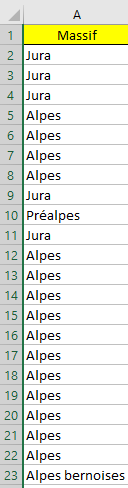
Comme dans les TABLES, on ne veut pas de doublon, il faut les éliminer et EXCEL peut nous aider.

Dans la barre du haut choisir « Données »



Et ensuite le bouton « Supprimer les doublons » , on passera de la situation ayant toutes les données, à la situation ayant les données mais sans doublon.

Avec doublons initialement : Sans doublon :



Faites cela sur toutes les données sauf celle des « Cols » qui sera la TABLE principale.

Pour les « lieux » en fait il faudra constituer **une seule colonne** puisque tous les lieux seront concentrés dans une seule TABLE.

Lieux de départ

…

…

Lieux d’arrivée

…

…

Lieux de départ

…

…

Lieux d’arrivée

…

…

Lieux

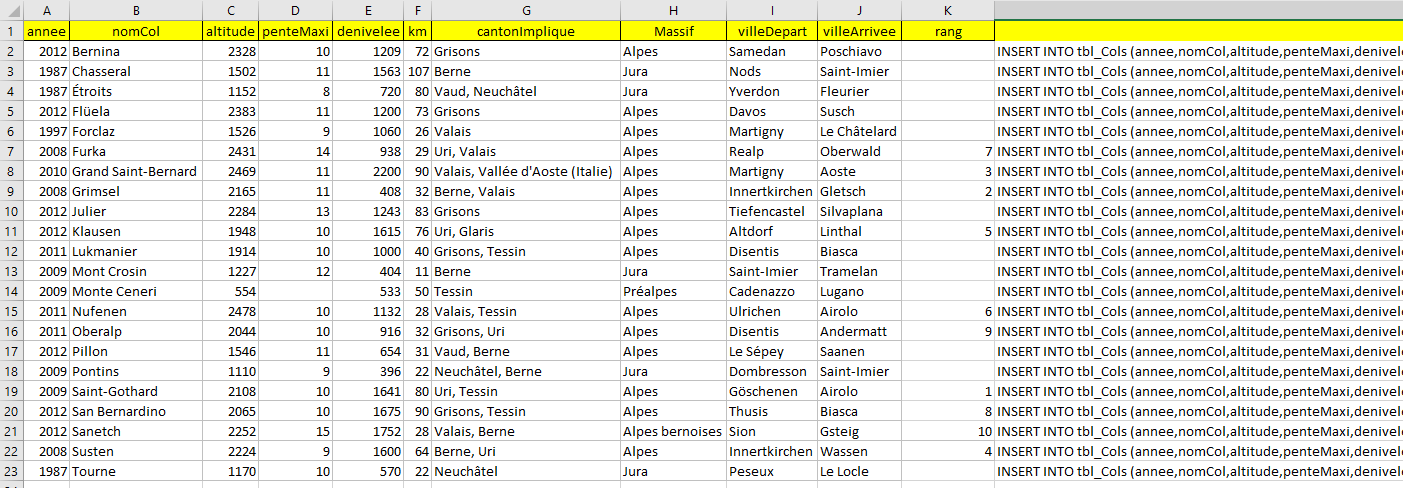
## Calculer les cellules de manière à obtenir le code SQL d’insertion

Il faut savoir que :

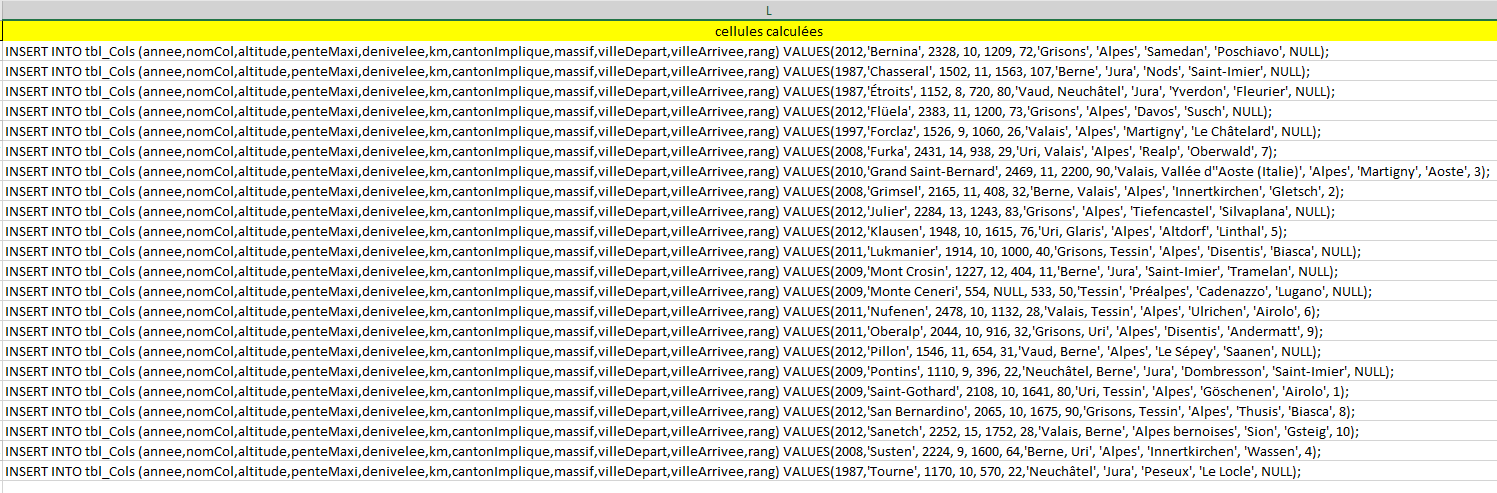
* Le caractère de concaténation est : &
* Dans la chaîne SQL INSERT INTO…, les valeurs « de type chaîne » sont à mettre entre apostrophes, exemple : ‘Neuchâtel’ mais pas les valeurs de nombres, exemple : 456.
* Pour « casser » une chaîne de caractères : Il faut concaténer, exemple : "Truc Muche" devient en 2 chaînes, "Truc" & " " & "Much" (& " " & pour créer un espace).

Dans EXCEL, pour la 1ère ligne, la chaîne SQL calculée devient :  
*(Attention vous n’avez peut-être pas les mêmes numéros de cellules !!!)* :

="**INSERT INTO** tbl\_Cols (annee,nomCol,altitude,penteMaxi,denivelee,km) **VALUES**("&A2&",'"&B2&"', "&C2&", "&D2&", "&E2&", "&F2&");"

Vous devez obtenir un tableau comme ceci pour la future TABLE « tbl\_Cols » : 

Suite du tableau sur la droite…



Faites la même chose pour les 4 autres TABLES.

Prendre des précautions pour pallier aux apostrophes dans la chaîne SQL

Il faut modifier les formules créées ci-dessus en intégrant une autre formule permettant de remplacer les apostrophes (') dans la chaîne par des doubles apostrophes ('') *(pas des guillemets (") !!!)*.

Si vous ne le faites pas, il sera impossible d’introduire une donnée texte comportant des apostrophes exemple : « Val d’Aoste »

*(pour SQL l’apostrophe est un début de chaîne, lorsqu’il en voit un dans la chaîne SQL le considérera comme la fin de la chaîne et dès lors la syntaxe ne sera plus correcte car il restera l’apostrophe de fin… bref ça buggera !).*

En conclusion : Dans une chaîne d’un ordre SQL, « Val d’Aoste » doit devenir « Val D’’Aoste ».

Il va de soi que le mécanisme n’est valable QUE pour la chaîne SQL mais que dans la base de données, il y aura « Val d’Aoste » inscrit.

Pour passer d’un apostrophe à un double apostrophes, la formule à intégrer pour chaque champ de « type chaîne » est :

**SUBSTITUE(champ à substituer;"'";"''")**

Prendre la précaution de mettre « NULL » pour une absence de valeur

Pour le champ « penteMaxi » il faudra mettre **NULL** pour les cols n’ayant pas de pente renseignée vu que ce champ est optionnel.

*(Pour un champ de « type texte », une absence de valeur est « une chaîne vide » :* ***""***

*Pour un champ de type « numérique », une absence de valeur est :* ***NULL****)*

**SI(ESTVIDE(champ à tester de type numérique);"NULL"; champ à tester de type numérique)**

## Constituer votre fichier d’insertion des données

Vous êtes prêt maintenant pour créer votre fichier avec toutes les données de chaque TABLE simplement par un copié-collé des cellules EXCEL dans un fichier NOTEPAD++ que vous pourrez nommer : « **X-164-Ex2\_Cols\_CH\_INSERT.sql** » .*(Vous pourrez ensuite intégrer le code de ce fichier au code de création pour n’en faire plus que un seul.)*

Après l’exécution du fichier de création des TABLES qui vous a été donné et de ce fichier d’insertion des données, vous êtes prêt pour la suite qui sera de **créer l’intégrité référentielle**, les clés étrangères (FK) et d’y insérer leurs données en cohérence avec celles des TABLES existantes.

## Contraintes sur les scripts de création (fourni) et d’insertion

(Votre script) Il faut pouvoir passer plusieurs fois le script d’insertion en effaçant les données et **remettant à 1 les compteurs au début de chaque script**. *(c.f. ALTER TABLE simple)*

(Le script fourni) Il pourra être passé plusieurs fois de suite sans erreur mais bien entendu il supprimera toutes les données déjà introduites et créera à nouveau les TABLES.

## Création des références et de l’intégrité référentielle

Comme nous avons créé nos TABLES avec chacune sa clé primaire (PK nommée « numero ») et issue d’un numéro automatique, il faudra faire correspondre le « numero »(PK) avec la clé étrangère (FK) dans le champ qui est déjà préparé (« num\_tbl…. »).

Pour ce faire il faut savoir qu’il est possible d’insérer une requête dans un ordre d’UPDATE *(mise à jour)*, comme on peut le faire pour un ordre INSERT également.

**Voici comment :**

**UPDATE** tbl\_Cols SET **num\_tblCantons =**(**SELECT** numero FROM tbl\_Cantons **WHERE** **tbl\_Cantons.cantonImplique=tbl\_Cols.cantonImplique**);

Suite à cet ordre **TOUS LES ENREGISTREMENTS** de la TABLE « tbl\_Cols » seront modifiés et le champ « num\_tblCantons » sera **renseigné** avec les bonnes valeurs numériques.

Lisant l’ordre SQL ci-dessus on peut dire ceci :

On met à jour le champ « num\_tblCantons » de la TABLE « tbl\_Cols » avec la valeur de la PK « numero » de la TABLE « tbl\_Cantons » appliquant la condition que les 2 champs **tbl\_Cantons.cantonImplique et tbl\_Cols.cantonImplique soient égaux.**

On comprend mieux maintenant laraison pour laquelle on avait conservé la valeur « sous forme de chaîne » du nom du canton dans la TABLE « tbl\_Cols ».

Une fois l’opération de mise à jour effectuée, on pourra supprimer le champs « **cantonImplique » de la TABLE « tbl\_Cols » puisque le lien avec la TABLE « tbl\_Cantons » sera fait par le biais du champ « num\_tblCantons ».**

**On pourra ainsi créer les clés étrangère (FK) entre « num\_tblCantons » de la TABLE « tbl\_Cols » et « numero » de la TABLE « tbl\_Cantons ».**

Faites-en de même pour les TABLES « tbl\_Massifs », « tbl\_Lieux » et « tbl\_CoupsCoeur ».

A la fin du fichier mettez ce bout de code que je n’explique pas encore mais qui pourra vous permettre de contrôler vos données par un affichage de celle-ci (**SELECT**…).

-- Contrôler par cette requête

**SELECT** annee,nomCol, tbl\_Cantons.cantonImplique,tbl\_Massifs.massif, tbl\_LieuxDepart.ville, tbl\_LieuxArrivee.ville, tbl\_CoupsCoeur.rang

**FROM** tbl\_Cols

**LEFT JOIN** tbl\_Cantons **ON** tbl\_Cols.num\_tblCantons = tbl\_Cantons.numero

**LEFT JOIN** tbl\_Massifs **ON** tbl\_Cols.num\_tblMassifs = tbl\_Massifs.numero

**LEFT JOIN** tbl\_Lieux AS tbl\_LieuxDepart **ON** tbl\_Cols.num\_tblLieuxDepart = tbl\_LieuxDepart.numero

**LEFT JOIN** tbl\_Lieux AS tbl\_LieuxArrivee **ON** tbl\_Cols.num\_tblLieuxArrivee = tbl\_LieuxArrivee.numero

**LEFT JOIN** tbl\_CoupsCoeur **ON** tbl\_Cols.num\_tblCoupsCoeur = tbl\_CoupsCoeur.numero

**ORDER BY** annee **ASC**, nomCol **ASC** ;

## Suppression des champs de la TABLE « tbl\_Cols » ayant servi à créer les références

Pour ce faire il faut modifier la structure de la table « tbl\_Cols ».

L’ordre de modification de la structure d’une TABLE est : **ALTER TABLE**…

Dans notre cas nous devons supprimer une colonne (champs) donc la syntaxe générique pour ce faire est :

**ALTER TABLE** **nom de la table** **DROP COLUMN** **nom de la colonne ;**

Il faut le faire sur tous les champs qui ne sont plus utiles dès maintenant.

## CREATION DES CLES ETRANGERES et de l'intégrité référentielle

De nouveau il faut modifier la structure de la TABLE « tbl\_Cols » donc il faut utiliser à nouveau : **ALTER TABLE…**

Nous devons ajouter une contrainte de clé étrangère, pour ce faire, utiliser : **ADD CONSTRAINT**

**ALTER TABLE nom de la table**

**ADD CONSTRAINT nom de la contrainte**

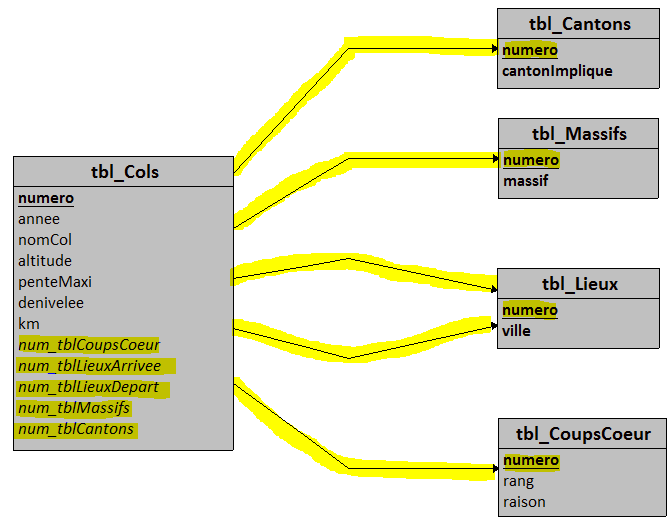
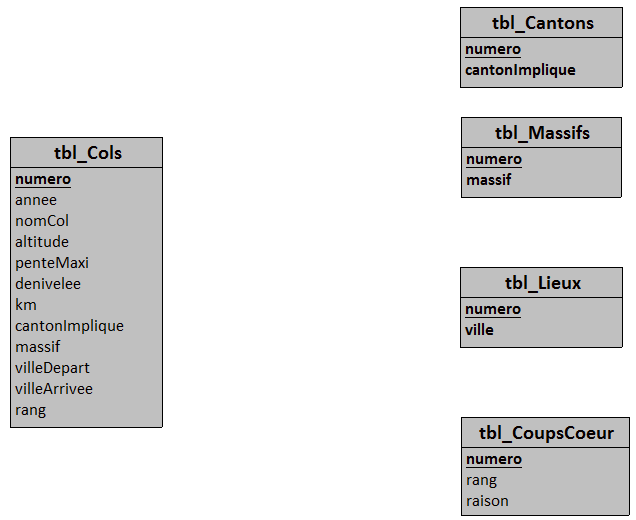
**FOREIGN KEY(colonne de la FK) REFERENCES nom de la table référencée (colonne de la PK);**

Il faut le faire sur toutes les clés étrangères.

A ce stade, les TABLES sont correctes avec l’intégrité référentielle

Voici ci-dessous la transformation que nous avons effectuée sur le modèle de données.

MLD au début du travail MLD final avec les données insérées



Champs utilisés temporairement

# Contrôle des DONNEES

## Contrôle des données de chaque TABLE

Vous pouvez copier-coller les lignes de codes ci-dessous pour visualiser toutes les données contenues dans les TABLES.

J’ai juste abrégé la « raison » du « coup de cœur » qui prendre beaucoup de place, c’est un texte long.

Une fonction (**LEFT(chaîne,nombre)**) me permet de prendre les 100 premiers caractères. Je concatène ensuite avec les « … » par le caractère de concaténation de chaîne en SQL qui est « || ».

**SELECT** \* **FROM** tbl\_Cantons;

**SELECT** \* **FROM** tbl\_Massifs;

**SELECT** \* **FROM** tbl\_Lieux;

**SELECT** numero AS "PK", rang AS "Rang", **LEFT(raison,100)||'...'** AS "Raison" **FROM** tbl\_CoupsCoeur;

**SELECT** numero AS PK,annee,nomCol,num\_tblCantons AS "FK Cantons", num\_tblMassifs AS "FK Massifs", num\_tblLieuxDepart AS "FK départ", num\_tblLieuxArrivee AS "FK arrivée", num\_tblCoupsCoeur AS "FK C.coeur",km, denivelee AS "Dénivelée", altitude AS "Altitude", penteMaxi AS "Pente max" **FROM** tbl\_Cols;

## Contrôle des données introduites par une requête

Il est judicieux à ce stade de bien contrôler que la requête donne les données correctes.

Sans plus d’explication (nous verrons les requêtes sélection plus tard dans ce cours) voici la requête mettant en jeu toutes les données utiles sauf le champ « raison » de la TABLE « tbl\_CoupsCoeur » qui est trop grand pour être visualisé sans altérer l’aspect visuel des données.

**SELECT** annee,nomCol, tbl\_Cantons.cantonImplique,tbl\_Massifs.massif, tbl\_LieuxDepart.ville, tbl\_LieuxArrivee.ville, tbl\_CoupsCoeur.rang FROM tbl\_Cols

**LEFT JOIN** tbl\_Cantons **ON** tbl\_Cols.num\_tblCantons **=** tbl\_Cantons.numero

**LEFT JOIN** tbl\_Massifs **ON** tbl\_Cols.num\_tblMassifs **=** tbl\_Massifs.numero

**LEFT JOIN** tbl\_Lieux **AS** tbl\_LieuxDepart **ON** tbl\_Cols.num\_tblLieuxDepart **=** tbl\_LieuxDepart.numero

**LEFT JOIN** tbl\_Lieux **AS** tbl\_LieuxArrivee **ON** tbl\_Cols.num\_tblLieuxArrivee **=** tbl\_LieuxArrivee.numero

**LEFT JOIN** tbl\_CoupsCoeur **ON** tbl\_Cols.num\_tblCoupsCoeur **=** tbl\_CoupsCoeur.numero

**ORDER BY** annee **ASC**, nomCol **ASC;**