SQL Déclaratif

Du standard à la pratique

Solutions au manuel d'exercices

ATTENTION : *Veuillez noter que les solutions proposées fourniront la réponse attendue par chacun des exercices mais qu'elles n'ont pas la prétention d'être les seules réponses possibles, bien au contraire ! De plus, les solutions ont été optimisées au mieux, mais il est possible que dans certains cas, d'autres solutions le soient encore plus.*

MODULE I :

DDL (Data Definition Language - Langage de définition de données)

Exercice 1.1 – La syntaxe des ordres suivants est-elle correcte ? Si non, pourquoi ?
Attention, les tables sont peut-être liées… !
N'hésitez pas à tester les requêtes directement !

```
1  CREATE TABLE T_office
2  ( office_id INTEGER,
3  office_address VARCHAR(30),
4  CONSTRAINT PK_office PRIMARY KEY (office_id))
5
6  CREATE TABLE T_course
7  ( crs_code CHAR(8) NOT NULL PRIMARY KEY,
8  crs name VARCHAR(30)
9  CONSTRAINT UK_crs UNIQUE (crs_name))
10
11 CREATE TABLE T_professor
12 ( prf_id INTEGER NOT NULL PRIMARY KEY,
13 prf_name VARCHAR(30),
14 prf_course CHAR(8)
15 CONSTRAINT PK_course REFERENCES T_course (crs_code),
16 ON DELETE SET NULL,
17 office_id CHAR(2) REFERENCES T_office,
18 CONSTRAINT prf_name UNIQUE (prf_name))
```

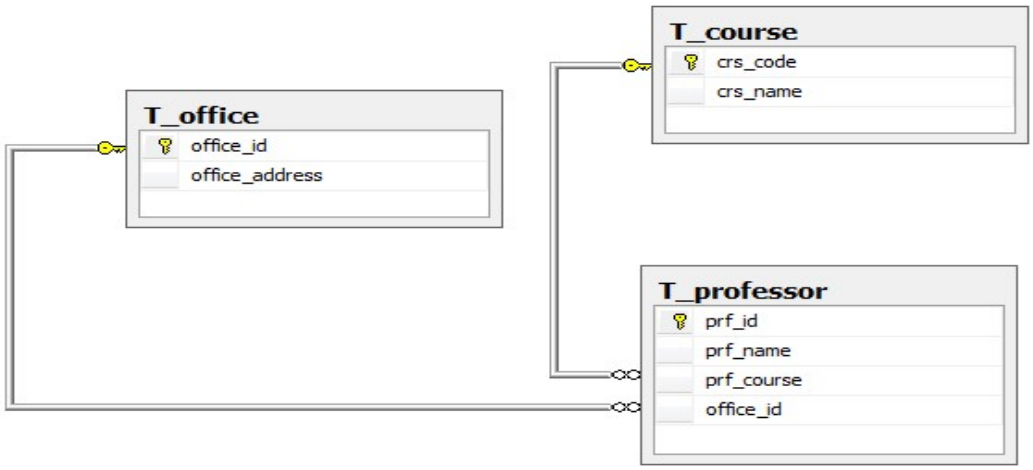
Solution :

- La première requête fonctionnera très bien telle qu'elle sous SQL-Server
- Deuxième requête :
    - Ligne 8 : il faut remplacer l'espace entre « crs » et « name » par un symbole « _ » (underscore)
    - Ligne 8 : il faut une virgule en fin de ligne, après le VARCHAR(30)
      Alternative : vous pouvez également supprimer « (crs_name) » en fin de ligne 9 et ne pas mettre de virgule en fin de ligne 8. Dans ce cas, la contrainte « UK_crs » de la ligne 9 sera une contrainte colonne qui portera implicitement sur la colonne « crs_name » de la ligne 8

```
CREATE TABLE T_course (
    crs_code CHAR(8) NOT NULL PRIMARY KEY,
    crs_name VARCHAR(30),
    CONSTRAINT UK_crs UNIQUE (crs_name)
)
```

- Troisième requête :
    - Ligne 15 : supprimer la virgule en fin de ligne car « ON DELETE SET NULL » fait partie de la contrainte
    - Ligne 17 : la colonne « office_id » qui fait référence à la colonne de même nom dans la table « T_office » précédemment créée doit être du même type que cette colonne, c'est-à-dire de type « INTEGER » et non « CHAR(8) »

```
CREATE TABLE T_professor (
    prf_id INTEGER NOT NULL PRIMARY KEY,
    prf_name VARCHAR(30),
    prf_course CHAR(8)
    CONSTRAINT PK_course REFERENCES T_course (crs_code)
    ON DELETE SET NULL,
    office_id INTEGER REFERENCES T_office,
    CONSTRAINT prf_name UNIQUE (prf_name)
)
```



Exercice 1.2 – **A partir des données présentées dans le tableau suivant, proposer le code de la table T_MAINTENANCE_MTN.**
Cette table devra contenir les 4 contraintes suivantes : contrainte de clé primaire, contrainte d'unicité, contrainte check et contrainte NOT NULL. Ces contraintes porteront sur 4 colonnes ou combinaisons de colonnes distinctes.

Jour	Machine	Numéro	Vitesse	Température	Heure	Evénement
Ven	Massicot	147			21 :18	Défaut de lame
Sam	Relieuse	63	16		16 :15	Arrêt pour maintenance
Jeu	Presse	87	6	62	11 :40	Bavure encre
Sam	Relieuse	79	16		17 :11	Reprise
Mer	Presse	89	6	55	08 :28	Recadrage
Mar	Presse	132	8	68	09 :58	Changement encre
Mer	Massicot	111			10 :17	Graissage coulisseau

Solution :

- Une contrainte colonne de type « check » est proposée sur la colonne « Jour » dont on remarque qu'elle ne contient que des noms de jours de la semaine écris sur 3 lettres
- Une contrainte colonne de type « NOT NULL » est proposée sur la colonne « Machine » dont on peut observer clairement qu'elle ne contient pas de valeur NULL
- Une contrainte table de clé primaire est proposée sur la colonne « Numéro » qui ne semble pas pouvoir contenir deux numéros identiques
- Une contrainte table d'unicité est proposée sur la combinaison des colonnes « Machine » et « Heure ». La combinaison des valeurs trouvées sur une même ligne dans ces deux colonnes semble ne jamais être identique

```
CREATE TABLE T_MAINTENANCE_MTN (
    MTN_JOUR        CHAR(3) CHECK (MTN_JOUR IN ('Lun','Mar','Mer','Jeu','Ven','Sam','Dim')),
    MTN_MACHINE     CHAR(8) NOT NULL,
    MTN_NUMERO      INTEGER ,
    MTN_VITESSE     INTEGER,
    MTN_TEMPERATURE INTEGER,
    MTN_HEURE       TIME,
    MTN_EVENEMENT   VARCHAR(100)
    CONSTRAINT PK_MAINTENANCE PRIMARY KEY (MTN_NUMERO),
    CONSTRAINT UK_MACHINE_HEURE UNIQUE (MTN_MACHINE,MTN_HEURE)
)
```

Exercice 1.3 – **Créer une table pour y stocker les produits à vendre, avec les rubriques suivantes : identifiant, référence magasin, référence fabricant, code EAN13, prix de vente. Cette table fera en outre référence aux tables T_TAUX_TVA, T_RAYON_RYN, T_FABRICANT_FBQ.**
Mettez en place toutes les contraintes nécessaires. La table produit contiendra au minimum les colonnes proposées, mais peut en contenir d'autres au besoin.

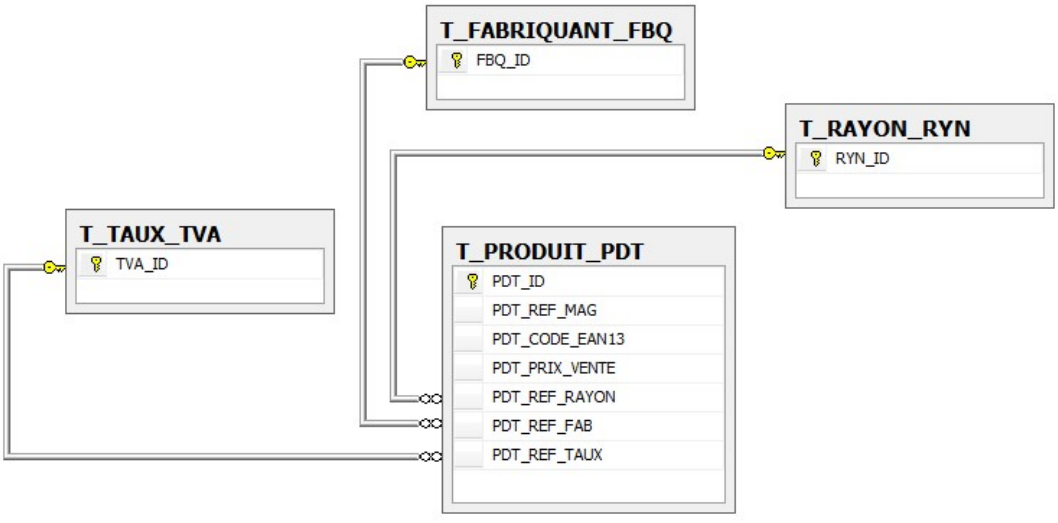
Solution :

```
CREATE TABLE T_RAYON_RYN (
    RYN_ID INT PRIMARY KEY
)

CREATE TABLE T_FABRIQUANT_FBQ (
    FBQ_ID INT PRIMARY KEY
)

CREATE TABLE T_TAUX_TVA (
    TVA_ID INT PRIMARY KEY
)

CREATE TABLE T_PRODUIT_PDT (
    PDT_ID          INT PRIMARY KEY,
    PDT_REF_MAG     VARCHAR(50),
    PDT_CODE_EAN13  VARCHAR(13),
    PDT_PRIX_VENTE  MONEY,
    PDT_REF_RAYON   INT,
    PDT_REF_FAB     INT,
    PDT_REF_TAUX    INT,
    CONSTRAINT FK_RAYON FOREIGN KEY (PDT_REF_RAYON)
        REFERENCES T_RAYON_RYN (RYN_ID),
    CONSTRAINT FK_FAB FOREIGN KEY (PDT_REF_FAB)
        REFERENCES T_FABRIQUANT_FBQ (FBQ_ID),
    CONSTRAINT FK_TAUX FOREIGN KEY (PDT_REF_TAUX)
        REFERENCES T_TAUX_TVA (TVA_ID)
)
```



<u>Exercice 1.4</u> – Soit le code de création de table repris ci-après et pour lequel les annotations suivantes concernant les fonctions utilisées pourront être utiles (sous Oracle, demander le script et des explications au formateur) :

- **« RTRIM(…) » et « LTRIM(…) » enlèves les espaces blancs respectivement à droite et à gauche de l'élément entre parenthèses**
- **« SUBSTRING(…,x,y) » renvoi la chaine de caractère commençant à « x » et se terminant « y » caractères après « x », à partir de la chaine de caractères donnée entre parenthèses**
- **« CONVERT(TYPE,…) » renvoi la valeur fournie dans le « TYPE » demandé**

```
1  CREATE TABLE T_VOITURE_VTR (
2      VTR_ID              INTEGER     NOT NULL PRIMARY KEY,
3      VTR_IMMATRICUL      CHAR(10)    NOT NULL UNIQUE,
4      VTR_CARBURANT       CHAR(2)     NOT NULL DEFAULT 'ES' CHECK(VTR_CARBURANT IN ('ES','GO','PL')),
5      VTR_PUISSANCE_FISC  INTEGER     NOT NULL CHECK(VTR_PUISSANCE_FISC BETWEEN 1 AND 20),
6      VTR_NB_PLACES       INTEGER     NOT NULL CHECK(VTR_NB_PLACES BETWEEN 1 AND 7),
7      VTR_MODELE          VARCHAR(20) CHECK(RTRIM(LTRIM(VTR_MODELE)) NOT LIKE ''),
8      VTR_CONSTRUCTEUR    VARCHAR(16) CHECK(RTRIM(LTRIM(VTR_CONSTRUCTEUR)) NOT LIKE ''),
9      VTR_NUMERO_SERIE    VARCHAR(25) NOT NULL CHECK(RTRIM(LTRIM(VTR_NUMERO_SERIE)) NOT LIKE ''),
10     CONSTRAINT CK_IMMATRICULATION CHECK(((CONVERT(INTEGER,SUBSTRING(VTR_IMMATRICUL,9,1)) BETWEEN 0 AND 9)
11                                          AND (SUBSTRING(VTR_IMMATRICUL,10,1) BETWEEN '0' AND '9')
12                                          AND (SUBSTRING(VTR_IMMATRICUL,9,2) < '96'))
13                                         OR ((CONVERT(INTEGER,SUBSTRING(VTR_IMMATRICUL,9,1)) = 2)
14                                             AND (SUBSTRING(VTR_IMMATRICUL,10,1) IN ('A','B')))),
15     CONSTRAINT CK_PUISS_PLACE     CHECK(VTR_NB_PLACES - 1 < VTR_PUISSANCE_FISC),
16     CONSTRAINT UK_MDL_CTR_NSR     UNIQUE (VTR_MODELE,VTR_CONSTRUCTEUR,VTR_NUMERO_SERIE)
17 )
```

Parmi les lignes suivantes, lesquelles seront refusées et pourquoi ?

ID	IMMA	CARB	PUISS	PLC	MDL	CONST	NUM_SERIE
14	'478 XDA 78'	'ES'	9	5	'305'	'PEUGEOT'	'00014578'
31	'1447 MD 44'	'ES'	'7'	5		'CITROEN'	'0001578'
7	'5475 MRT 91'	'GO'	5	4	'204'	'PEUGEOT'	'0001474578'
11	'1744 BC 76'	'GO'	7	5			'00025678'
15	'4412 LR 75'	'GO'	7	4	'305'	'PEUGEOT'	'00014578'
17	'971 VTR 96'		7	5	'306'	'PEUGEOT'	'00017548'
19	'991 SDT 75'	'ES'	8	5	'MEGANE'	'RENAULT'	'00014578'
20	'991 SDT 75'	'ES	5	4	'MEGANE'	'RENAULT'	'00014578'
14	'4875 ZT 94'		7	5		'RENAULT'	'005784'
7	'5474 MRT 91'	'GPL'	5	4	'PT CRUISER'	'CHRYSLER'	'0000050214'

<u>Explication du code :</u>

- **Ligne 2 :** la colonne « VTR_ID » de type est la clé primaire de la table et n'accepte que des valeurs entières
- **Ligne 3 :** la colonne « VTR_IMMATRICUL » ne peut contenir que des chaines de caractères longues de 10 caractères maximum. Cette colonne est obligatoire et les valeurs sont uniques

- **Ligne 4 :** la colonne « VTR_CARBURANT » ne peut contenir que deux caractères. Sa valeur ne peut pas être « NULL » et si on tente tout de même d'y entrer cette valeur, le système remplira le champ avec la valeur « ES » par défaut. Les valeurs entrées doivent correspondre à l'une des valeurs suivantes : « ES », « GO » ou « PL »
- **Ligne 5 :** les valeurs de la colonne « VTR_PUISSANCE_FISC » doivent être des entiers, les valeurs sont obligatoires et comprises entre 1 et 20
- **Ligne 6 :** les valeurs de la colonne « VTR_NB_PLACES » doivent être des entiers, les valeurs sont obligatoires et comprises entre 1 et 7
- **Ligne 7 :** les valeurs de la colonne « VTR_MODELE » sont des chaines de caractères longues d'au maximum 20 caractères contenant au moins une lettre (une valeur remplie uniquement d'espaces blancs ne sera pas acceptée). Les valeurs NULL sont acceptées
- **Ligne 8 :** même chose que pour « VTR_MODELE » avec des chaines de caractères d'au maximum 16 caractères
- **Ligne 9 :** même chose que pour « VTR_MODELE » avec des chaines de caractères d'au maximum 25 caractères mais cette fois, les valeurs NULL ne sont pas acceptées
- **Lignes 10 à 14 :** il s'agit d'une contrainte qui vérifie que les caractères 9 et 10 de la chaine de caractère entrée pour la colonne « VTR_IMMATRICUL » sont soit compris entre 01 et 95 (lignes 10 à 12), soit égaux à « 2A » ou « 2B » (lignes 13 et 14)
- **Ligne 15 :** contrainte qui vérifie si la valeur de « VTR_PUISSANCE_FISC » est strictement plus grande que la valeur de « VTR_NB_PLACES » moins 1
- **Ligne 16 :** contrainte d'unicité sur la combinaison des champs « VTR_MODELE » « VTR_CONSTRUCTEUR » et « VTRE_NUMERO_SERIE »

Solution :

(Nous considérons que les blancs dans la tables sont des valeurs « NULL » et pas des chaines de caractères remplis d'espaces blancs, auquel cas les insertions 2, 4 et 9 ne fonctionneraient pas, dû aux contraintes « CHECK » posées sur les colonnes « MDL » et « CONST », respectivement nommées « VTR_MODELE » et « VTR_CONSTRUCTEUR » dans le code de la table)

- **Insertion n°1 :** fonctionne sans encombre
- **Insertion n°2 :** fonctionne également ; les guillemets simples autour du '7' dans la colonne « PUISS » signifient qu'on essaye d'entrer la chaine de caractères '7' et non le chiffre « 7 », mais le système fera lui-même la conversion et cela ne créera pas d'erreur
- **Insertion n°3 :** la chaine de caractères entrée dans la colonne « IMMA » est trop longue d'un caractère. Il vaut mieux laisser les deux derniers chiffres et enlever l'un des caractères précédents, dû à la contrainte « CK_IMMATRICULATION » (lignes 10 à 14 du code de la table)
- **Insertion n°4 :** fonctionne sans encombre
- **Insertion n°5 :** la combinaison des données contenues dans les colonnes « MDL », « CONST » et « NUM_SERIE », c'est-à-dire « 305, PEUGEOT, 00014578 » existe déjà à la première insertion. La contrainte définie en ligne 16 du code de la colonne empêche les doublons au niveau de cette combinaison de données
- **Insertion n°6 :** les deux derniers caractères de la colonne « IMMA » étant plus grands que 95, l'insertion est refusée. Notons que ne rien insérer dans la colonne « CARB » ne change rien

puisque par défaut, si aucune valeur n'est fournie pour cette colonne, le système y insèrera la valeur « ES » (ligne 4 du code de la table)

- **Insertion n°7 :** fonctionne sans encombre ; les valeurs de la colonne « NUM_SERIE » ne doivent pas être uniques
- **Insertion n°8 :** doublon au niveau de la colonne « IMMA » et de la combinaison « MDL-CONST-NUM_SERIE »
- **Insertion n°9 :** doublon au niveau de la colonne « ID »
- **Insertion n°10 :** doublon au niveau de la colonne « ID » ; valeur trop longue pour la colonne « IMMA » ; valeur « GPL » refusée au niveau de la colonne « CARB » : selon la contrainte de la ligne 4 du code de la table, les seules valeurs acceptées dans cette colonne sont « ES », « GO » ou « PL »

Exercice 1.5 – Deux scripts vous sont fournis : « DBSlide_LoadDB.sql » et « DBSlide_LoadData.sql ».

Créer une base de données que l'on appellera « DBSlide ». Tenter d'exécuter les scripts fournis… Cela ne devrait pas fonctionner. A vous de les corriger !

Solution :

Pour corriger le script « DBSlide_LoadDB.sql », il faut bien avoir en tête le principe de parents-enfants entre les tables : les tables parents doivent être créées avant les tables enfants. En analysant les contraintes de clés étrangères de chacune des tables, il apparait que les tables doivent être crééent dans l'ordre suivant : SECTION => STUDENT => PROFESSEUR => COURSE ; la table GRADE peut être créée à tout moment puisqu'elle n'est ni parent ni enfant d'aucune table). Il faut ensuite corriger les erreurs de logique suivantes :

- Dans la table « grade », la contrainte check porte sur une valeur nommée « value ». Elle doit porter sur la colonne à laquelle la contrainte est associée. Il faut donc remplacer « value » par « grade », le nom de la première colonne de la table
- Dans la table « course », la colonne « professor_id » est de type « char(1) ». Mais cette colonne fait partie de la clé étrangère liant la table « course » à la table « professor », dans laquelle la colonne « professor_id » référencée est de type « int ». Cela rend le référencement impossible. Il faut donc changer le type de la colonne « professor_id » de la table « course » en « int »
- La clé étrangère de la table « student » fait référence à une colonne « section » de la table « section » qui n'existe pas. Il faut bien entendu faire référence à la colonne « section_id »

Pour corriger le script « DBSlide_LoadData.sql », il faut de nouveau respecter les règles imposées par les relations parents-enfants entre les tables. En se référant au script « DBSlide_LoadDB.sql », on remarque facilement les contraintes de clés étrangères qui impliquent que :

- La table « section » est parente des tables « student » et « professor ». Les données de la table « section » doivent donc exister avant qu'il n'en existe dans ses tables enfants. De la même manière, lors de la suppression des données, il faudra que les données des tables « student » et « professor » soient effacées avant d'effacer celles de la table « section »
- La table « professor » est parente de la table « course », ses données doivent donc exister avant d'essayer d'en faire apparaitre dans la table « course ». Inversement, lors de la suppression, les données de la table « course » devront disparaitre avant de faire disparaitre les données de la table « professor »
- La table « grade » n'est liée à aucune table, ses données peuvent donc être supprimées ou modifiées à n'importe quel moment

Lors de la suppression des données : GRADE => COURSE => PROFESSOR => STUDENT => SECTION

Lors de l'insertion de données : SECTION => PROFESSOR => STUDENT => COURSE => GRADE

Il faut en outre corriger les problèmes suivants :

- Les lignes insérées dans la table « professor » n'insèrent des données que dans 7 colonnes, or selon le code de création de la table, celle-ci en attend 8. Il faut rajouter à chaque insertion de ligne une valeur pour la première colonne de la table : « professor_id »
- La dernière ligne insérée dans la table « professor » (concernant « François Louveaux ») essaye d'inscrire ce professeur dans une section « 1110 » inexistante dans la table parent « section ». Comme certains étudiants sont également inscris dans cette section et que personne n'est inscrit dans la section « 1111 », modifions la valeur de cette « section_id » dans la table « section » par « 1110 »
- La dernière ligne insérée dans la table « course » (concernant le cours « EING2383 ») fait référence à un professeur dont l'id serait « 8 », mais ces valeurs, dans la table « professor » ne vont que de 1 à 6. Il faut donc choisir une valeur de « professeur_id » existante
- La première ligne insérée dans la table « grade » renseigne une valeur « N/A » trop longue pour cette colonne dont le type est CHAR(2). De plus, la valeur insérée dans cette colonne doit correspondre à l'une des valeurs acceptées par la contrainte CHECK de la table (par exemple « IG » qui est la seule valeur manquante parmi les lignes insérées)

Exercice 1.6 – **Une fois les scripts de l'exercice précédent corrigés, les tables créées et remplies, réaliser les modifications suivantes :**

- **Autoriser la table « SECTION » à accepter des valeurs NULL pour la colonne « delegate_id »**
- **Ajouter à la table « SECTION » une clé étrangère faisant pointer la colonne « delegate_id » vers la colonne « student_id » de la table « STUDENT »**
- **Supprimer la colonne « course_id » de la table « STUDENT »**
- **Faire en sorte que les données de la colonne « student_id » de la table « STUDENT » soient auto-incrémentées**
- **En ne supprimant aucune donnée, modifier le type de la colonne « section_id » de la table « section » afin qu'il soit en CHAR(4). Cela impliquera peut-être d'autres modifications…**

Solution :

```
1  ALTER TABLE section ALTER COLUMN delegate_id int NULL
2
3  ALTER TABLE section
4  ADD CONSTRAINT FK_section_student FOREIGN KEY (delegate_id)
5      REFERENCES student (student_id)
6
7  ALTER TABLE student DROP COLUMN course_id
8
9  ALTER TABLE section DROP CONSTRAINT FK_section_student
10 ALTER TABLE student DROP CONSTRAINT PK_student
11 ALTER TABLE student DROP COLUMN student_id
12 ALTER TABLE student ADD student_id int identity(1,1)
13                         CONSTRAINT PK_student PRIMARY KEY
14 ALTER TABLE section
15 ADD CONSTRAINT FK_section_student FOREIGN KEY (delegate_id)
16     REFERENCES student (student_id)
17 -- Il est impossible en code T-SQL de modifier l'ordre des colonnes
```

```
19  ALTER TABLE student DROP CONSTRAINT FK_student_section
20  ALTER TABLE professor DROP CONSTRAINT FK_professor_section
21  ALTER TABLE section DROP CONSTRAINT PK_section
22  ALTER TABLE section ALTER COLUMN section_id char(4) NOT NULL
23  ALTER TABLE student ALTER COLUMN section_id char(4) NOT NULL
24  ALTER TABLE professor ALTER COLUMN section_id char(4) NOT NULL
25  ALTER TABLE section ADD CONSTRAINT PK_section PRIMARY KEY (section_id)
26  ALTER TABLE student
27  ADD CONSTRAINT FK_student_section FOREIGN KEY (section_id) REFERENCES section
28  ALTER TABLE professor
29  ADD CONSTRAINT FK_professor_section FOREIGN KEY (section_id) REFERENCES section
```

Exercice 1.7 – Améliorer le script « DBSlide_LoadDB.sql » afin qu'il commence par supprimer les tables, pour ensuite les recréer sans leurs clés étrangères. Une fois chaque table créée, leur rajouter les clés étrangères

Solution :

(Se référer par exemple aux lignes 3 à 5 de la solution de l'exercice précédent pour la syntaxe d'ajout des clés étrangères à chacune des tables ; rajouter l'ordre « DROP TABLE nom_table » en début de script pour chacune des tables, en respectant la règle parents-enfants (supprimer les tables enfants avant de supprimer les tables parents))