Bases de données - Langage SQL

1. Présentation générale

1.1. Une base de données, qu'est ce que c'est ?

Ce que dit le programme : "L'administration, les banques, les assurances, les secteurs de la finance utilisent des bases de données, systèmes d'informations qui stockent dans des fichiers les données nombreuses qui leur sont nécessaires. Une base de données relationnelle permet d'organiser, de stocker, de mettre à jour et d'interroger des données structurées volumineuses utilisées simultanément par différents programmes ou différents uti lisateurs. Un logiciel, le système de gestion de bases de données (SGBD), est utilisé pour la gestion (lecture, écriture, cohérence, actualisation...) des fichiers dans lesquels sont stockées les données. L'ac cès aux données d'une base de données relationnelle s'effectue en utilisant un langage informatique qui permet de sélectionner des données spécifiées par des formules de logique, appelées requêtes d'interro gation et de mise à jour. L'objectif est de présenter une description applicative des bases de données en langage de requêtes SQL (Structured Query Language). Il s'agit de permettre d'interroger une base présentant des données à travers plusieurs relations."

1.2. Structure d'une table : vocabulaire

Une base de données est un ensemble de plusieurs tableaux à deux dimensions (penser à des tableurs sur Excel) dans lequel les données sont organisées et que nous appelerons **tables**. Concentrons nous tout d'abord sur une seule table avant de comprendre comment nous les mettrons en relation les unes avec les autres.

Exemple Observons le fichier ECG2.csv correspondant à la table donnant des informations sur les élèves de la classe :

Id	Nom	Classe	LV1	LV2	Option 1	Option 2
1	AYACHE Clarisse	ECG2B	ANGLAIS	ALLEMAND	APPLIQ	ECON.SOC.& HIST. MC
2	AYADI Rafik	ECG2A	ANGLAIS	ESPAGNOL	APPLIQ	HIST.GEO.GEOPOLITIQ.
3	BAPTISTA Élise	ECG2B	ESPAGNOL	ANGLAIS	APPLIQ	ECON.SOC.& HIST. MC
4	BARSALON Constance	ECG2B	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
5	BENCHAHBOUNE Florent	ECG2A	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
6	CAUCHY Flora	ECG2A	ESPAGNOL	ANGLAIS	APPLIQ	HIST.GEO.GEOPOLITIQ.
7	CESSIEUX Mathis	ECG2A	ANGLAIS	ITALIEN	APPLIQ	ECON.SOC.& HIST. MC
8	CHADUIRON Axel	ECG2B	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
9	CHAPUIS Emma	ECG2B	ANGLAIS	ESPAGNOL	APPRO	ECON.SOC.& HIST. MC
10	CISERON Clara	ECG2B	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
11	COUDERC Mathieu	ECG2A	ANGLAIS	ESPAGNOL	APPLIQ	HIST.GEO.GEOPOLITIQ.
12	DEMARS Fantine	ECG2B	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
13	DEVILLE Jules	ECG2A	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
14	DIDIER Gabriel	ECG2A	ANGLAIS	ESPAGNOL	APPRO	ECON.SOC.& HIST. MC
15	DISTINGUIN Eliot	ECG2B	ANGLAIS	ALLEMAND	APPLIQ	ECON.SOC.& HIST. MC
16	FOUILLAT Zoé	ECG2B	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
17	GAILLARD Ilona	ECG2B	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
18	GARNAUD Achille	ECG2A	ANGLAIS	ITALIEN	APPLIQ	ECON.SOC.& HIST. MC
19	GORRINDO Meline	ECG2A	ANGLAIS	ALLEMAND	APPLIQ	ECON.SOC.& HIST. MC
20	GOUTELLE Lizéa	ECG2A	ANGLAIS	ESPAGNOL	APPLIQ	HIST.GEO.GEOPOLITIQ.
21	HAMOUDA Naïm	ECG2A	ANGLAIS	ALLEMAND	APPLIQ	HIST.GEO.GEOPOLITIQ.
22	HERZI Elies	ECG2A	ANGLAIS	ESPAGNOL	APPLIQ	HIST.GEO.GEOPOLITIQ.
23	HOMEYER Vladimir	ECG2A	ANGLAIS	ALLEMAND	APPRO	HIST.GEO.GEOPOLITIQ.
24	HOUEL Antoine	ECG2B	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
25	INGLESE Cameron	ECG2A	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
26	ISSA Lena	ECG2A	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
27	JACQUEMOND Clémence	ECG2A	ANGLAIS	ALLEMAND	APPLIQ	ECON.SOC.& HIST. MC
28	KAROUI Anis	ECG2A	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
29	KREMER Louis	ECG2A	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
30	LAVASTRE Johan	ECG2A	ANGLAIS	ESPAGNOL	APPRO	ECON.SOC.& HIST. MC
31	LEAUTE Ronan	ECG2A	ESPAGNOL	ANGLAIS	APPLIQ	HIST.GEO.GEOPOLITIQ.
32	LEVET Siméo	ECG2A	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
33	MENZER Younes	ECG2B	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
34	MOREAU Valentin	ECG2B	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
35	MOREL Clara	ECG2A	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
36	MUGNIER Clara	ECG2A	ANGLAIS	ALLEMAND	APPLIQ	ECON.SOC.& HIST. MC
37	RABHI Samy	ECG2A	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
38	RICHARD Lilou	ECG2A	ANGLAIS	ITALIEN	APPLIQ	HIST.GEO.GEOPOLITIQ.
39	ROBERT Alicia	ECG2A	ANGLAIS	ALLEMAND	APPLIQ	ECON.SOC.& HIST. MC
40	ROUSTAN–FIORI Nina	ECG2A	ANGLAIS	ESPAGNOL	APPLIQ	HIST.GEO.GEOPOLITIQ.
41	SIMONET Théophile	ECG2A	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
42	SOUMAILI Fany Moinahamissy	ECG2A	ANGLAIS	ESPAGNOL	APPLIQ	HIST.GEO.GEOPOLITIQ.
43	SZABO-REBUT Tali	ECG2B	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
44	TARDY Charlotte	ECG2A	ANGLAIS	ESPAGNOL	APPLIQ	HIST.GEO.GEOPOLITIQ.
45	TOFFA Grâce	ECG2A	ANGLAIS	ALLEMAND	APPLIQ	HIST.GEO.GEOPOLITIQ.
46	VEZZARO Roman	ECG2A	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
47	VINCENT Aymeric	ECG2B	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC
48	ZAEME EL KAMOUNI Manal	ECG2A	ESPAGNOL	ANGLAIS	APPLIQ	HIST.GEO.GEOPOLITIQ.
49	ZOUAOUI Chérif	ECG2A	ANGLAIS	ESPAGNOL	APPLIQ	ECON.SOC.& HIST. MC

- Chaque ligne de la table est appelée **enregistrement**. Si un nouvel élève veut intégrer la classe, on procède à un nouvel enregistrement et l'ordre de ces enregistrements n'a pas d'importance.
- Pour chaque enregistrement, on doit remplir les différents **champs** correspondant aux différentes "cases" de notre table.
- Dans chaque **colonne**, l'ensemble des types de données des différents champs s'appelle le **domaine**. Le programme nous dit de nous cantonner à des domaines d'entiers (**INT**) ou de chaines de caractères (**TEXT**), même s'il existe d'autres domaines (décimaux, dates, booléens...) **NULL** sera utilisé pour les cases vides.
- Les entités sont les éléments de la population que l'on étudie. L'ensemble des colonnes d'une entité s'appelle les **descripteurs** de l'entité.
- On appelle **schéma de la table** le descriptif de la table, constitué d'un nom (celui de la table), suivi de la liste de ses colonnes et de leur domaine.
- **SQL** (**Structured Query Language**) est un langage permettant de faire des requêtes sur des bases de données.
- La lecture et la modification de bases de données est permise par un logiciel, le système de gestion des bases de données (**SGBD : Système de Gestion des Bases de Données Relationnel**), qui interprète le langage SQL.

1.2.1. Remarque

- D'habitude les commandes du langage SQL sont mises en majuscule (SELECT, INSERT, ...), ce n'est pas obligatoire parce que le langage SQL n'est pas sensible à la casse mais c'est une pratique que l'on prendra pour des questions de lisibilité.
- Les noms des colonnes seront indiqués usuellement en minuscules et dépendront de chaque base de données. On travaillera cette année principalement avec des colonnes comme id, nom, adresse, prix...

★ *Exercice 1 (Appropriation du vocabulaire)*

1. Combien de colonnes y a-t-il dans la table ECG2 ? Lister les.
2. Combien d'enregistrements y a-t-il dans la base de données ECG2 ?
3. Quel est le domaine de la colonne Nom ?
4. Donner le schéma de cette table.
5. Donner l'enregistrement relatif à soi.

2. Bases de données - Introduction

2.1. L'intérêt

Les bases de données permettent d'enregistrer de manière structurée un grand nombre de données. Elles facilitent l'accès et la maintenance des données. Le cours se concentre sur les bases de données **relationnelles** qui sont le modèle le plus répandu, en particulier pour enregistrer les informations des sites internet (listes d'utilisateurs, commandes, etc). Nous découvrirons également le langage de requête SQL, également très répandu sur internet de nos jours, qui permet d'interroger ces bases de données.

2.2. Une mise en situation

Vous avez décidé d'ouvrir une pizzeria sur Saint-Etienne qui fonctionne uniquement sur livraison. Vous êtes heureux de votre petite entreprise qui propose une large gamme de pizzas et embauche déjà plusieurs livreurs. Vous disposez également de quelques véhicules (3 scooters et 1 voiture) pour assurer les livraisons. Bien sûr vous vous devez de bien garder note de toutes les commandes passées sinon les clients pourraient ne pas être livrés ou alors vous ne pourriez plus faire vos comptes à la fin de la semaine.

2.3. Ce qu'il ne faut pas faire

Pour vous organiser, vous vous procurez un ordinateur et réalisez un petit fichier dans un tableur :

Client	Adresse	Pizza	Livreur	Véhicule	Heure	Livraison
Dumbledore	28 av. de la libération	Canicatti	Harry	Scooter	19 :00	19 :45
Weasler	Chemin de traverse	Royale	Hermione	Voiture	18 :00	18 :30
McGonagall	4 rue Privet Drive	Forezienne	Ron	Scooter	12 :00	-
Weasler	Chemin de traverse	Savoyarde	Harry	Scooter	12 :00	12 :30
Riddle	Allée des embrumes	végétarienne	Harry	Voiture	14 :00	-
Dumbledore	28 avenue de la libération	Océane	Hermione	Scooter	14 :00	14 :30

Au départ les choses fonctionnent bien mais petit à petit vous vous rendez compte des limites du système.

- à chaque commande il vous faut retaper toutes les informations sur le client comme son adresse. On risque alors de faire une faute de saisie ou encore de saisir l'adresse différemment ce qui va créer des incohérences (exemple de l'adresse de Dumbledore).
- Un jour vous vous rendez compte que Riddle s'appelle en fait Voldemort. Pour corriger votre base de données il faut alors chercher dans toutes les lignes (et il peut y en avoir beaucoup), la présence de Riddle et les modifier. On voit donc qu'il y a trop de redondance d'information dans ce tableau.
- Lors de la livraison par Harry de la pizza savoyarde chez Weasley, il se produit un problème : en réalité il y a deux Weasley (Fred et Georges) qui vivent au chemin de traverse et Harry se trompe de destinataire. On a encore un problème d'incohérence puisque le même nom Weasley correspondait en fait à deux individus distincts.
- Vous constatez qu'un des scooters a un rétroviseur cassé, mais le tableau ne vous permet pas de distinguer les scooters, vous ne savez donc pas quel livreur interroger.

Au bout d'un mois, vous n'en pouvez plus et vous jetez l'éponge : il vous faut un bien meilleur système d'organisation des données.
Les **bases de données relationnelles** sont un paradigme d'organisation des données qui per met de **maximiser la cohérence** des informations tout en **minimisant la redondance**. Il est alors bien plus facile de lire, ajouter ou modifier des données.

3. Définitions et vocabulaire

3.1. Principes

Au lieu d'utiliser un tableur global, une **base de données relationnelle** est constituée d'un ensemble de "petites" **tables** qui permettent de mieux organiser les données.
Nous allons comprendre **la notion essentielle de clé** pour savoir comment faire comprendre à la machine comment ces petites tables sont liées entre elles.

Voici un exemple de table qui contient les informations concernant les clients :

Client			
idClient	**Nom**	**Prénom**	**Adresse**
1	Dumbledore	Albus	28 av. de la Libération
2	Weasley	Fred	Chemin de traverse
3	McGonagall	Minerva	4 rue Privet Drive
4	Riddle	Tom	Allée des embrumes
5	Weasley	Georges	Chemin de traverse
6	Diggory	Cédric	Le terrier

On constate l'ajout d'une information `idClient` qui est un numéro permettant d'identifier de manière unique et éviter ainsi les confusions. On appelle `idClient` la **clé primaire** de la table Client.

Ajoutons maintenant des tables `Produit`, `Livreur` et `Vehicule` :

Produit		
idProduit	**Designation**	**Prix**
1	Savoyarde	13.00
2	Royale	9.00
3	Forézienne	12.00
4	Bretonne	11.00
5	Océane	12.00
6	Végétarienne	11.00
7	Canicatti	13.00

Livreur		
idLivreur	**Nom**	**Prénom**
1	Granger	Hermione
2	Potter	Harry
3	Weasley	Ron

Vehicule		
idVehicule	**Type**	**Kms**
1	Scooter	25000
2	Voiture	42000
3	Scooter	17000
4	Scooter	5000

On constate que chacune de ses tables possède sa propre **clé primaire**.

Maintenant il est intéressant de voir comment créer une nouvelle table dans laquelle on pourra enfin gérer les commandes :

Commande							
id	**idClient**	**idProduit**	**idLivreur**	**idVehicule**	**Date**	**Heure**	**Livraison**
1	1	7	2	4	13-04	19 :00	19 :45
2	5	2	1	1	13-04	18 :00	18 :30
3	3	3	3	4	14-04	12 :00	NULL
4	5	1	2	1	14-04	12 :00	12 :30
5	4	6	2	2	14-04	14 :00	NULL
6	1	5	1	4	14-04	14 :00	14 :30

Dans la table "Commande", nous constatons qu'en plus de la clef primaire *id* correspondant à un numéro unique de commande, les clients, produits, livreurs et véhicules sont désignés grâce à leurs identifiants respectifs **de leur propre table**. On appelle cela des **clefs étrangères**. Les clefs étrangères sont indispensables pour relier les informations des différentes tables. Nous découvrons également la possibilité d'utiliser la valeur `NULL` si besoin ; pour nous il s'agit des commandes qui n'ont pas encore été livrées.
Comparons maintenant cette méthode d'organisation avec la précédente

- Lors du renseignement d'une nouvelle commande, il est inutile de devoir réécrire toutes les informations sur le client, le produit, le livreur, etc. Seuls les numéros d'identification suffisent.
- Pour changer le nom *Riddle* en *Voldemort* il n'y a qu'une seule table à lire et une seule case à considérer.
- On ne risque plus de confondre les homonymes car chaque entité est identifée par sa clef primaire.

3.2. Définitions et vocabulaire : rappels et compléments

Nous avons maintenant compris qu'une **base de données relationnelle** est un ensemble de tables ayant des relations les unes avec les autres. On se propose maintenant de donner une définition plus *mathématique* de la notion de table :
Une **table** `Table(colonne_1, ..., colonne_n)`, également appelée **relation**, est un ensemble fini d'**enregistrements**, c'est-à-dire de $\boldsymbol{n}$**−uplets** $(x_1,...,x_n)$ où chaque $x_i \in D_i$ représente la valeur de la **colonne** correspondante. L'ensemble D_i est appelé **domaine** de la colonne. Tous les éléments d'un même domaine sont du même **type**. On utilisera les types suivants : `TEXT` pour les chaines de caractères, `INT` pour les entiers, et les cases vides sont repérées par "`NULL`".

Nous constatons que le monde des bases de données utilise un vocabulaire spécifique mais il ne faut pas perdre de vue l'essentiel : une relation est en fait un tableau. Fréquemment on utilisera le vocabulaire plus intuitif suivant :

Vocabulaire bases de données	Vocabulaire intuitif
Relation	Table
colonne	Colonne
Enregistrement / n−uplet	Ligne
Domaine	Type

On appelle **schéma** de la table sa structure, c'est à dire l'ensemble ordonné de ses colonnes et les domaines associés. On le note souvent : `Table(colonne_1 DOMAINE_1, ..., colonne_n DOMAINE_n)`
L'ensemble des schémas de toutes les tables est appelé **schéma de la base de données**.

Exemple

Le schéma de la base de données de la pizzéria est le suivant :
`Client(idClient ENTIER, Nom TEXTE, Prénom TEXTE, Adresse TEXTE)`
`Produit(idProduit ENTIER, Désignation TEXTE, Prix FLOTTANT)`
`Livreur(idLivreur ENTIER, Nom TEXTE, Prénom TEXTE)`
`Véhicule(idVéhicule ENTIER, Type TEXTE, Kms ENTIER)`
`Commande(id ENTIER, idClient ENTIER, idProduit ENTIER, idLivreur ENTIER, idVéhicule ENTIER, Date DATE, Heure HEURE, Livraison HEURE)`

Nous avons décidé ici de souligner les clés primaires et d'écrire en italique les clés étrangères.
Dans une base de données, il se peut que des colonnes de tables différentes portent le même nom. Pour lever cette ambiguité, on utilisera la notation `Table.colonne`.
Dans l'exemple, deux colonnes `Nom` existent : `Client.Nom` et `Livreur.Nom`

4. Langage de requête SQL

4.1. Vocabulaire

Les bases de données sont consultables et gérables grâce à des logiciels spécifiques. En classe, nous utliserons DB Browser for SQLite. Pour nos TPs, une base de données sera représentée sous forme de fichier .sqlite. Pour interroger et modifier des bases de données, on dit que l'on formule des **requêtes**. Les requêtes sont formulées dans un langage de requête spécifique, le langage SQL.

4.2. À savoir avant de commencer

- le langage SQL ne se préoccupe pas de la casse. Mais pour plus de lisibilité, on prendra l'habitude d'écrire en majuscules les mots clés du langage et en minuscule le reste (c'est à dire les noms de tables, de colonnes, ou d'entrées)
- contrairement à Python, l'indentation ou le saut de ligne n'ont pas d'importance. On passera parfois à la ligne et on indentera pour plus de lisibilité
- les commentaires sont précédés de -
- les différentes requêtes sont séparées par des **points-virgules**.

4.3. Les requêtes SQL à connaître : Lecture d'une table, avec ou sans jointure

Dans le jargon, on dit que l'on **interroge** la table.

4.3.1. SELECT ... FROM ...

```
SELECT colonne1, colonne2, colonne3 FROM base
```

Permet d'afficher l'intégralité du contenu de certaines colonnes d'une base de donnée `table`. Cela renvoie uniquement les colonnes sélectionnés, sous forme d'un tableau contenant une colonne par colonne sélectionnée en ignorant les autres colonnes non sélectionnées.
Si on veut obtenir toutes les colonnes d'une base de données, on écrit SELECT *.

4.3.2. SELECT ... FROM ... WHERE ...

```
SELECT colonne1, colonne2, colonne3 FROM base WHERE <condition>
```

Permet d'obtenir le même résultat que dans le paragraphe précédent en filtrant uniquement les enregistrements qui répondent à certaines conditions logiques.
On pourra utiliser avec profit les opérateurs `AND`, `OR` ou `NOT`, =, <, >, <=, >=, +, - et *.
⚠ Attention, la syntaxe de "différent de" n'est pas la même que dans python. En SQL, on utilisera <>. On remarque également que le signe = n'est pas doublé comme dans python pour réaliser des conditions.

Exemple

`SELECT Prénom FROM Client WHERE Name='Weasley'`
est par exemple une requête qui permettra de retrouver les prénoms des membres de la famille Weasley dans la base de données `Client`. On remarquera qu'il faut encadrer les chaines de caractère avec des apostrophes.

4.3.3. SELECT ... FROM ... INNER JOIN ... ON ...

```
SELECT * FROM base1 INNER JOIN base2 ON base1.colonne_n = base2.colonne_p
```

permet de joindre deux bases de données `base1` et `base2`, c'est-à-dire croiser les informations des deux bases. On précise comment joindre ces bases avec les conditions logiques qui suivent `ON` : seront renvoyés toutes les colonnes des deux tables pour tous les enregistrements dont la colonne `colonne_n` de `base1` est égale à la colonne `colonne_p` de `base2`.
Cette requête étant longue donc peu lisible, on pourra l'écrire sur plusieurs lignes pour plus de lisibilité :

```
SELECT *
FROM base1
INNER JOIN  base2
ON base1.colonne_n = base2.colonne_p
```

★ *Exercice 2 (Ouverture d'une BDD existante)*

1. Chercher sur cahier de prépa, télécharger et enregistrer dans votre espace personnel le fichier `pizza.sqlite`.
2. Ouvrir le programme `DB Browser for SQLite` sur l'ordinateur.
3. À l'aide du bouton '`Ouvrir une base données`', ouvrir la base de données `pizza.sqlite`.
4. Parcourir les onglets suivants pour se familiariser avec l'interface :
   - '`Structure de la Base de Données`' permet de voir la liste des tables.
     Développer les petits triangles '▶' pour voir le schéma de chaque table.
   - '`Parcourir les données`' permet de voir le contenu de chaque table. On tombe tout d'abord sur la table '`Client`'. Visualiser également les 4 autres tables.

★ *Exercice 3 (Interrogation d'une BDD existante : première requêtes)*
La base de données ouverte dans l'exercice précédent correspond à l'exemple présenté dans le document de cours. Pour toutes les questions suivantes, proposer une requête SQL. On pourra tester les résultats de la requête dans l'onglet '`Exécuter le SQL`'. L'onglet se divise en deux parties : une pour taper les requêtes SQL l'autre pour observer les résultats sous forme de tables. On utilisera le bouton "Exécuter la ligne courante" (▶|) ou "Exécuter tout" (▶) pour lancer l'évaluation d'une requête.

1. Obtenir la liste des prénoms des clients.
2. Obtenir la listes des pizzas qui coûtent moins de 10 euros.
3. Obtenir la listes des pizzas dont le prix est entre 10 euros et 12 euros inclus.
4. Obtenir le prix de la pizza végétarienne.
5. Obtenir la liste de tous les scooters qui ont parcouru plus de 10 000 kms.
6. Obtenir la liste des commandes qui n'ont pas été livrées (heure de livraison `NULL`).
7. Obtenir la liste des noms de pizzas commandés par le client de code `idClient 1`.
8. Obtenir la liste des noms et prénoms des livreurs ayant livré une pizza chez le client de code `idClient 1`.

4.4. Les requêtes SQL à connaître : Création d'une table

4.4.1. CREATE TABLE ...

```
CREATE TABLE matable(colonne1 <TYPE>, colonne2 <TYPE>, colonne3 <TYPE>)
```

permet de créer une table vide. Il faudra dans un second temps la remplir avec des enregistrements.
On pourra préciser `PRIMARY KEY` derrière le type de la colonne qui jouera le rôle de clé primaire.

4.4.2. INSERT INTO ...

```
INSERT INTO matable VALUES('valeur1','valeur2','valeur3')
```

permet d'effectuer un enregistrement. Si on veut effectuer plusieurs enregistrements d'un coup, on utilisera la commande suivante :

```
INSERT INTO matable VALUES('valeur1a','valeur2a','valeur3a'),('valeur1b','valeur2b','valeur3b')
```

★ *Exercice 4*

Nous voulons créer une base de données des notes d'élèves. Chaque élève a un numéro étudiant, un prénom et un nom. Les devoirs ont un titre, une matière et un numéro d'identification unique. Il est alors possible de représenter cette base de donnée avec 3 tables.

```
Eleves(idEleve INTEGER, Prenom TEXT, Nom TEXT)
Devoirs(idDevoir INTEGER , Titre TEXT, Matiere TEXT)
Resultats(idEleve INTEGER, idDevoir INTEGER, note INTEGER, rang INTEGER)
```

1. Dans le schéma de base de données proposé, indiquer les colonnes qui correspondent à des clefs primaires. Identifier les clefs étrangères.
2. À l'aide des boutons 'Nouvelle base de donnée', 'Créer table', construire cette base de données. Le programme affichera alors automatiquement les commandes SQL permettant de créer les tables, mais pour la table Devoirs, rentrer le code "à la main" dans l'onglet 'Exécuter le SQL'. (Dans tous les cas, on pensera à mentionner les clefs primaires).
3. Dans l'onglet 'Parcourir les données', à l'aide du bouton 'Nouvel enregistrement', alimenter les tables Eleves et Resultats à l'aide des informations ci-dessous. Vous choisirez vous-même les numéros d'identifiants des élèves. Pour la table Devoirs, rentrer les enregistrements "à la main".

Notes au DS 3 (Maths) : analyse

Jack	Sparrow	4
Peter	Pan	2
Sherlock	Holmes	16
Thomas	Anderson	20
Ellen	Ripley	17

Notes au DS 7 (Anglais) : vocabulary

Jack	Sparrow	20
Peter	Pan	18
Sherlock	Holmes	10
Thomas	Anderson	4
Ellen	Ripley	15

4. Dans l'onglet 'Exécuter le SQL', afficher la liste des prénoms des élèves.
5. Afficher la liste des matières traitées.
6. Afficher la liste des notes obtenues.
7. Afficher la liste des matières où au moins un éléve a eu moins de 3.
8. Afficher la liste des noms des élèves ayant eu moins de 10 au devoir numéro 3.

4.5. Les requêtes SQL à connaître : Modification d'une table

4.5.1. DELETE FROM ...

`DELETE From matable` permet de supprimer tous les enregistrements d'une table

`DELETE From matable WHERE <condition>` permet de supprimer tous les enregistrements d'une table vérifiant une condition

4.5.2. UPDATE ...

`UPDATE table SET colonne='valeur'` permet de remplacer toutes les entrées de la colonne 'colonne' par 'valeur'.

`UPDATE table SET colonne='valeur' WHERE <condition>`
permet de remplacer toutes les entrées de la colonne "colonne" vérifiant la condition par 'valeur'.

★ *Exercice 5*

1. Thomas Anderson veut maintenant se faire appeler Neo -tout court-. Modifier la base de données en conséquence.
2. Puisqu'il est arrivé des bricoles à Ellen Ripley, elle ne fait plus partie de la classe. Modifier les bases de données en conséquence pour supprimer les données faisant référence à elle.