TP 6

Bases de données

1 Modèle relationnel **2**
1.1 Vocabulaire 2
1.2 Clé primaire 3
1.3 Clé étrangère 3

2 Langage `SQL` **4**
2.1 Commandes exigibles 4
2.2 Commandes non exigibles 8

3 Exercices **11**

Compétences attendues.

- ✓ Connaître le vocabulaire des bases de données.
- ✓ Connaître la notion de clé primaire et de clé étrangère.
- ✓ Savoir créer une table.
- ✓ Savoir sélectionner des données, les insérer, les supprimer ou les modifier.
- ✓ Utiliser les opérateurs logiques pour sélectionner des données.
- ✓ Savoir réaliser une jointure.

Liste des commandes `SQL` exigibles aux concours.

- `CREATE TABLE entity` : Création d'une table.
- `SELECT attribut FROM entity WHERE condition` : Sélection de données dans une table vérifiant une condition.
- `INSERT INTO entity` : Insertion de données dans une table.
- `DELETE FROM entity WHERE condition` : Suppression de données d'une table vérifiant une condition.
- `UPDATE entity SET attribut="valeur" WHERE condition` : Mise à jour de données d'une table.
- `SELECT * FROM entity1 INNER JOIN entity2` : Réalisation d'une jointure.

Anthony Mansuy
Professeur de Mathématiques en deuxième année de CPGE filière ECG au Lycée Clemenceau (Reims)
Page personnelle : `http://anthony-mansuy.fr`
E-mail : `mansuy.anthony@hotmail.fr`

De nombreux secteurs ont besoin pour leur activité de traiter un grand nombre de données. Une base de données est un système organisé de ces informations. Les progrès de l'informatique ont permis de développer des logiciels capables de stocker ces bases de données de manière structuré et de les interroger pour en extraire efficacement des informations utiles, ce sont les *Système de Gestion des Bases de Données Relationnel* (SGBDR). Le langage le plus communément utilisé pour interroger les SGBDR est le langage SQL (*Structurel Query Langage*) dont nous allons présenter les rudiments.

1 Modèle relationnel

1.1 Vocabulaire

On considère des éléments d'une même **entité** sur lesquelles on a des informations (ces entités peuvent être des clients, des objets, des villes...).

Exemple. Quatre clients d'un fournisseur de desserts sont identifiés par leur `Id_client` numéroté $1, 2, 3, 4$:

- Le client dont l'identifiant est 1 s'appelle Dupond, il faut lui livrer 35 tiramisus à Reims.
- Le client dont l'identifiant est 2 s'appelle Le Gall, il faut lui livrer 30 crêpes à Paris.
- Le client dont l'identifiant est 3 s'appelle Prigent, il faut lui livrer 25 tartes à Reims.
- Le client dont l'identifiant est 4 s'appelle Gallois, il faut lui livrer 40 tiramisus à Lille.

L'entité peut être nommée ici "Client", les données portent sur leurs identifiants, leurs noms, les produits à livrer, leurs quantités ainsi que le lieu de livraison.

Définition.

- Les propriétés dont on a la valeur pour chaque élément d'une entité et qui forment les données s'appellent des **attributs**. Leur ensemble forme les **descripteurs** de l'entité.
- L'ensemble des valeurs possibles d'un attribut s'appelle le **domaine** de l'attribut. Ces valeurs doivent être toutes de même type (pour nous, essentiellement : `"INT"` (entiers) ou `"TEXT"` (chaînes de caractères)).

Exercice. Donner la liste des attributs de l'entité "Client" et les domaines de ces attributs.

Définition.

- On appelle **schéma de relation** le descriptif de la relation, constitué d'un nom (on prend souvent celui de l'entité), suivi de la liste de ses attributs et de leur domaine.
- Pour une entité donnée, la liste des valeurs pour chaque attribut s'appelle un **enregistrement**.
- On appelle **relation** l'ensemble des données pour tous les éléments. Ces données peuvent être organisées et présentées sous la forme d'une **table** à qui on donne un nom.

Exercice.

1. Donner le schéma de relation de l'entité "Client".

2. Donner les enregistrements pour chaque client de l'entité "Client".

Exemple. On peut ainsi représenter la table associée à l'entité "Client" :

Client

Id_client	Nom	Ville	Produit	Quantité
1	Dupond	Reims	Tiramisu	35
2	Le Gall	Paris	Crêpe	30
3	Prigent	Reims	Tarte	25
4	Gallois	Lille	Tiramisu	40

Remarque. Du fait de cette présentation très pratique, on confondra souvent les termes **relation** et **table**, **attribut** et **colonne**, **enregistrement** et **ligne**.

1.2 Clé primaire

Définition.

Une **clé primaire** d'une table est un attribut (ou une colonne) qui permet d'identifier de manière unique les entités de la table.

Exemple. L'attribut `Id_client` est une clé primaire alors que l'attribut `Ville` ne l'est pas (l'identifiant client permet de repérer le client alors que la ville, non).

Remarque. Dans cet exemple, la clé primaire est un attribut artificiel (`Id_client`) qui est incrémenté à chaque nouveau client. En effet, choisir l'attribut `Nom` est une mauvaise idée : si l'on doit plus tard ajouter des clients à la table, il n'est pas impossible que deux d'entre eux aient le même nom.

1.3 Clé étrangère

Définition.

- On appelle **clé étrangère** d'une table toute colonne qui référence la clé primaire d'une autre table. Cette clé étrangère met donc en relation cette table avec la première.
- On appelle **schéma relationnel** la donnée des schémas de tables (ou de relations) ainsi que les clés primaires et étrangères.

Remarque. L'usage est souvent de souligner les clés primaires et de placer un dièse (#) devant les clés étrangères.

Exemple. Considérons la table `Dessert` suivante, toujours utilisée par le même fournisseur de desserts :

`Dessert`

Produit	Prix	Stock	Gluten
Crêpe	0,40	200	Oui
Tarte	1,20	160	Oui
Tiramisu	1,80	140	Non

La colonne `Produit` est une clé primaire de la table `Dessert` et c'était une colonne de la table `Client`. Ces deux tables sont donc en relation et `Produit` est une clé étrangère de la table `Client`.

Exercice. Donner le schéma relationnel associé au modèle relationnel client-produit.

2 Langage SQL

Dans ce paragraphe, les règles générales seront énoncées pour une table (ou relation) nommée `entity`. On nommera `attribut1`, `attribut2`, etc. ses colonnes.

Le langage `SQL` ne fait pas la différence entre majuscule et minuscule. L'usage consiste cependant à écrire en `MAJUSCULE` les mots-clés du langage et en `minuscule` le reste. Les différentes requêtes sont séparées par des points-virgules.

2.1 Commandes exigibles

Propriété 1 (Création d'une table et déclaration des clés)

La requête

```
CREATE TABLE entity1 (
     attribut1 nom_du_domaine,
        ...
     attributi nom_du_domaine,
        ...
     attributj nom_du_domaine,
        ...
     attributn nom_du_domaine,
     PRIMARY KEY (attributi),
     FOREIGN KEY (attributj) REFERENCES entity2(attributj')
);
```

permet de :

- créer la table `entity1` dont les attributs sont `attribut1`, ..., `attributn` ;
- déclarer `attributi` comme clé primaire de la table `entity1` à l'aide du mot-clé `PRIMARY KEY` ;
- déclarer `attributj` comme clé étrangère de la table `entity1` qui référence la clé primaire `attributj'` de la table `entity2`.

Exemple. Donner des requêtes `SQL` pour définir les tables `Client` et `Dessert`.

Propriété 2 (Sélection de données dans une table)

(1) **Projection :** La requête

```
SELECT attribut1 FROM entity
```

commande l'affichage de la colonne correspondant à `attribut1` de la table `entity`.

On sélectionne plusieurs colonnes en les séparant par des virgules. On les sélectionne toutes en commandant :

```
SELECT * FROM entity
```

(2) **Sélection :** La requête

```
SELECT attribut1 FROM entity WHERE condition
```

commande l'affichage de la colonne correspondant à `attribut1` de la table `entity` en ne retenant que les lignes vérifiant la condition définie après `WHERE`.

La condition est un bouléen, elle s'énonce à l'aide des opérateurs de comparaison habituels : $<$, $<=$, $>$, $>=$, $<>$ (différent), $=$ (le signe égal n'est pas doublé), et on peut utiliser aussi les opérations arithmétiques $+$, $-$ et $*$.

Remarque. Il est possible de combiner plusieurs conditions à l'aide des opérateurs logiques `AND` (les deux conditions doivent être vérifiées), `OR` (au moins l'une des deux conditions doit être vérifiée) et `NOT` (la condition ne doit pas être vérifiée).

Exercice.

1. Donner une requête `SQL` pour obtenir les noms et les villes de la table `Client`.

2. Donner une requête `SQL` pour obtenir les noms des clients ayant commandé des tiramisus ainsi que les villes à livrer.

3. Donner une requête `SQL` pour obtenir les noms des clients ayant commandé des tiramisus mais qui ne sont pas à livrer à Lille.

Propriété 3 (Modification d'une table)

(1) **Insertion :** La requête

```
INSERT INTO entity VALUES(élément1, élément2, ...)
```

insère une ligne dans la table `entity`, dont les valeurs sont : `élément1` pour le champ correspondant à `attribut1`, `élément2` pour le champ correspondant à `attribut2`, ...

(2) **Suppression :** La requête

```
DELETE FROM entity WHERE condition
```

supprime les lignes dans la table `entity` vérifiant la `condition` définie après `WHERE`. La requête

```
DELETE FROM entity
```

supprime la table `entity` en entier.

(3) **Modification :** La requête

```
UPDATE entity SET attribut="valeur" WHERE condition
```

remplace la valeur à l'intersection de la ligne désignée par `condition` et de la colonne correspondant à `attribut`.

Exercice.

1. On souhaite ajouter un nouveau client : M. Lepeltier, habitant à Nancy et à qui il faut livrer 20 quatre-quarts. Donner une requête `SQL` permettant de l'ajouter à la table `Client`.

2. Donner une requête `SQL` permettant de supprimer de la table `Client` ceux dont les commandes ne dépassent pas 30 unités.

3. M. Dupond déménage à Strasbourg. Donner une requête `SQL` permettant de modifier la ville de M. Dupond dans la table `Client`.

Lorsque deux tables sont en relation via une de leur colonne, il est possible de faire apparaître l'ensemble des données des deux tables sur un seul tableau en réalisant une jointure.

Propriété 4 (Jointure)

Soit deux tables `entity1` et `entity2` telles que la colonne `attributk` de `entity1` et la colonne `attributk'` de `entity2` aient au moins une valeur commune. La requête

```
SELECT * FROM entity1 INNER JOIN entity2 ON entity1.attributk = entity2.attributk'
```

commande l'affichage, à la suite, des lignes de la table `entity1` et de la table `entity2`, pour les seuls cas où les champs de la colonne `attributk` sont égaux à ceux de la colonne `attibutk'`.

Remarque. Dans cette requête, toutes les colonnes ont été commandées par défaut (du fait de l'astérisque après `SELECT`). On peut, bien entendu, n'afficher que les colonnes qui nous intéresse en remplaçant l'astérisque par le nom des colonnes, séparées par des virgules (projection). On peut même imposer des conditions sur les lignes (sélection ou restriction).

Exercice.

1. Donner une requête `SQL` permettant d'afficher à la suite les lignes des tables `Client` et `Dessert`.

2. Donner une requête `SQL` permettant d'obtenir les noms des clients, les quantités, les prix et les stocks.

3. Donner une requête `SQL` permettant d'obtenir le prix du dessert de M. Prigent.

2.2 Commandes non exigibles

Les commandes suivantes ne sont pas exigibles mais, selon les termes du programme officiel, elles pourront être utilisées par commodité et si besoin.

Propriété 5 (Commandes `UNION`, `INTERSECT`, `EXPECT`)

(1) **Commande `UNION` :** La requête

```
SELECT attribut1 FROM entity1
UNION
SELECT attribut2 FROM entity2
```

commande l'affichage des données de la colonne correspondant à `attribut1` de la table `entity1` et celles de la colonne correspondant à `attribut2` de la table `entity2`, sous la forme d'une seule colonne.

(2) **Commande `INTERSECT` :** La requête

```
SELECT attribut1 FROM entity1
INTERSECT
SELECT attribut2 FROM entity2
```

commande l'affichage des données communes à la colonne correspondant à `attribut1` de la table `entity1` et à la colonne correspondant à `attribut2` de la table `entity2`.

(3) **Commande `EXPECT` :** La requête

```
SELECT attribut1 FROM entity1
EXPECT
SELECT attribut2 FROM entity2
```

commande l'affichage des données de la colonne correspondant à `attribut1` de la table `entity1` privées de celles de la colonne correspondant à `attribut2` de la table `entity2`.

Remarques.

1. On peut commander plusieurs colonnes (à condition d'en commander le même nombre pour les deux tables).
2. On peut ajouter des conditions pour chaque sélection (à l'aide de `WHERE`).

Exercice.

1. Donner une requête `SQL` pour obtenir les noms et les villes où livrer des tiramisus ainsi que les noms et les produits correspondant à une livraison inférieure à 30 unités.

2. Donner deux requêtes `SQL` (une avec `INTERSECT` et une avec `EXPECT`) pour obtenir les produits livrés à Reims qui sont sans gluten.

Propriété 6 (Fonctions d'agrégation)

La requête

```
SELECT SUM(attribut) FROM entity
```

commande l'affichage de la somme des données de la colonne correspondant à `attribut` de la table `entity` (si toutefois son type est numérique).

On peut remplacer `SUM` par :

- `MAX` : renvoie le maximum des données de la colonne ;
- `MIN` : renvoie le minimum des données de la colonne ;
- `AVG` : renvoie la moyenne (pour "average") des données de la colonne ;
- `COUNT` : renvoie le nombre de données de la colonne.

Remarque. On peut ajouter des conditions pour chaque sélection (à l'aide de `WHERE`).

Exercice.

1. Donner une requête `SQL` pour obtenir le total des quantités commandées.

2. Donner une requête `SQL` pour obtenir le total des desserts en stock contenant du gluten.

3. Donner une requête `SQL` pour obtenir le prix le plus élevé des desserts.

4. Donner une requête `SQL` pour obtenir la moyenne des quantités commandées par les clients.

5. Donner une requête `SQL` pour obtenir le nombre de clients ayant commandé plus de 30 unités.

Propriété 7 (Commandes `DISTINCT` et `ORDER BY`)

(1) **Commande `DISTINCT` :** La requête

```
SELECT DISTINCT attribut FROM entity
```

commande l'affichage des données de la colonne correspondant à `attribut` en évitant les doublons.

(2) **Commande `ORDER BY` :** La requête

```
SELECT attribut1 FROM entity ORDER BY attribut2
```

commande l'affichage des données de la colonne correspondant à `attribut1` en les classant dans l'ordre croissant correspondant à `attribut2`.

Remarques.

1. On peut commander plusieurs colonnes (à condition d'en commander le même nombre pour les deux tables).
2. On peut ajouter des conditions pour chaque sélection (à l'aide de `WHERE`).

Exercice.

1. Donner une requête `SQL` pour obtenir les différentes villes de la table `Client`.

2. Donner une requête `SQL` pour classer les noms des clients en commençant par ceux qui ont commandé le moins.

3 Exercices

Les étudiants disposant de leur ordinateur personnel pourront installer "DB Browser for SQLite". C'est une interface graphique permettant d'interroger une base de données avec des requêtes SQL. Le programme est disponible ici : `https://sqlitebrowser.org/dl/` . Vous pouvez choisir la version windows 64 zip(no installer) du programme.

Exercice 1 (★)

On considère la relation `Salle` suivante :

`Salle`

Numéro	Capacité	Ordinateur	Accès
302	40	Oui	Clé
310	45	Oui	Clé
312	60	Non	Badge
323	40	Oui	Clé

1. (a) Donner les attributs de la relation `Salle`.
   (b) Donner le domaine de chaque attribut de la relation.
   (c) Pour chaque attribut, dire s'il peut jouer le rôle de clé primaire dans la relation `Salle`.
2. Combien cette relation compte-t-elle d'enregistrements ?
3. Écrire le schéma de relation associé à la relation `Salle`.

Exercice 2 (★)

On considère les deux tables suivantes :

`Livre`

Réf	Id_auteur	Titre	Dispo	État
Re12	713	Poil de Carotte	Oui	Bon
Re14	713	L'écornifleur	Non	Bon
Ay9	27	La Vouivre	Non	Usé
Gi3	541	La Menteuse	Oui	Bon

`Écrivain`

Id_auteur	Nom	Prénom	Naissance
27	Aymé	Marcel	1902
541	Giraudoux	Jean	1882
713	Renard	Jules	1864

1. (a) Dans quel état sont les livres de Jules Renard ?
   (b) Quelle est la clé primaire de la table `Livre` ?
2. Quelle est la clé étrangère de la table `Livre` qui permet de la relier à la table `Écrivain` ?
3. Donner le schéma relationnel associé aux tables `Livre` et `Écrivain`.

Exercice 3 (★★)

On considère les trois tables suivantes :

Voiture

Plaque	Propriétaire	Marque	Modèle	Atelier
DG 103 HY	3	Peugeot	807	Carrosserie
EF 334 GA	1	Renault	Clio	Pneu
EI 189 KA	4	Fiat	Uno	Pneu
FA 934 TH	5	Lancia	Delta	Mécanique
BB 512 IJ	6	Renault	Twingo	Pneu
FD 246 MT	2	Fiat	500	Mécanique

Client

Id_client	Nom	Prénom	Téléphone
1	Perha	Marie	0668543452
2	Ridy	Anne	0675463309
3	Leclerc	Louis	0667240908
4	McGregor	Thomas	0679856423
5	Ricciardo	Constance	0619798669
6	Lauquie	Faustine	0638899821

Tarif

Atelier	Chef	Prix
Pneu	Paul	75
Mécanique	Roger	120
Carrosserie	Gérard	145

1. (a) Qui va s'occuper de la voiture de Louis Leclerc ?
   (b) De combien de Renault Paul va-t-il s'occuper ?
2. Au vu des relations entre les différentes entités représentées par ces trois tables :
   (a) Donner les clés primaires de chacune de ces trois tables.
   (b) Identifier des clés étrangères sur la table `Voiture` référençant les clés primaires des deux autres tables.
   (c) Écrire alors le schéma relationnel associé à ces trois tables.
3. Écrire les commandes `SQL` permettant de créer ces trois tables.
4. Que permettent d'effectuer les requêtes `SQL` suivantes :
   (a) `UPDATE Tarif SET Chef="Yves" WHERE Chef="Paul"`
   (b) `SELECT Marque,Modèle FROM Voiture WHERE Atelier="Pneu"`
   (c) `SELECT Propriétaire FROM Voiture WHERE Marque="Renault" AND Atelier="Pneu"`
   (d) `DELETE FROM Voiture WHERE Plaque="DG 103 HY"`
5. Écrire les requêtes `SQL` permettant de :
   (a) Sélectionner les colonnes `Plaque` et `Atelier` de la table `Voiture`.

(b) Sélectionner les noms des propriétaires d'une Peugeot (on veut une seule requête).

(c) Sélectionner les ateliers dont le taux horaire HT ne dépasse pas 100 euros.

(d) Sélectionner les numéros de téléphone des propriétaires de Renault venus pour un problème de pneu.

(e) Sélectionner les marques et les modèles des voitures réparées aux ateliers Carrosserie et Mécanique.

(f) Modifier le numéro de téléphone de Mme Ridy (son nouveau numéro est le 0651096754).

(g) La voiture de M. McGregor est réparée. Supprimer la ligne correspondante dans la table `Voiture`.

Exercice 4 (★★)

On s'intéresse dans cet exercice aux communes, départements et régions françaises, dont les informations sont rassemblées dans une base de données nommée `TP6-ex4` disponible sur mon site `anthony-mansuy.fr/` .

Il s'agit de donner des requêtes `SQL` pour interroger cette base de données afin de répondre aux questions posées ci-dessous.

1. Requêtes simples.

   (a) Déterminer la liste des régions.

   (b) Déterminer le nom et le numéro de département des communes.

   (c) Déterminer la liste des communes de plus de 100000 habitants.

   (d) Déterminer la liste des communes de la Marne de plus de 100000 habitants.

2. Fonctions d'agrégation.

   (a) Donner le nombre de communes répertoriées dans la base.

   (b) Écrire une requête donnant simultanément les populations de la commune la plus peuplée et de la commune la moins peuplée.

3. Jointures.

   (a) Donner la liste des communes avec le nom du département correspondant.

   (b) Donner la liste des communes avec les noms du département et de la région associés.

   (c) Donner la population moyenne des communes de la région Champagne-Ardenne.

4. Requêtes composées.

   (a) Donner la liste des communes dont la population dépasse celle de la commune la plus peuplée des Ardennes.

   (b) Donner la liste des communes dont la population est supérieure de la moyenne des populations des communes françaises.