Bases de données

Capacités attendues

- ✓ Identifier les concepts définissant le modèle relationnel.
- ✓ Savoir distinguer la structure d'une base de données de son contenu.
- ✓ Repérer des anomalies dans le schéma d'une base de données.
- ✓ Identifier les services rendus par un SGBD : persistance des données, gestion des accès concurrents, efficacité de traitement des requêtes, sécurisation des accès.
- ✓ Identifier les composants d'une requête.
- ✓ Construire des requêtes d'interrogation à l'aide des clauses du langage SQL : SELECT, FROM, WHERE, JOIN.
- ✓ Construire des requêtes d'insertion et de mise à jour à l'aide de : UPDATE, INSERT, DELETE.

1 Généralités

Exercice 1 – *(Création d'une table et import des données)*

1. Nous avons besoin d'un logiciel particulier pour naviguer dans une base de données et la gérer. Il en existe plusieurs, nous utilisons ici *DB Browser for SQLite* : `https://sqlitebrowser.org/`

   → Télécharger la version portable du logiciel ici et l'exécuter.

2. Dans le logiciel *DB Browser for SQLite*, créer une nouvelle base de données appelée `db_world.db`. Une fenêtre surgit, appuyer sur *annuler*. Notre base de données a été créée (on peut voir qu'il y a zéro table dedans, entre autres).

3. Cliquer sur l'onglet *Exécuter le SQL* puis recopier le code suivant dans la zone *SQL 1*. L'exécuter en cliquant sur le petit triangle situé au-dessus de la fenêtre *SQL 1* (ou en appuyant sur F5).

   Dans le langage SQL, la **fin d'une instruction est indiquée par un point-virgule**, et les **indentations sont décoratives** :

```
CREATE TABLE Countries (
    id INT,
    code CHAR(2),
    name VARCHAR,
    continent VARCHAR);
```

   Ce code permet de créer une **relation** (table) nommée `Countries`, dont les `attributs` (colonnes) sont :

   - `id` : un entier (`INT`);
   - `code` : une chaîne de 2 caractères (`CHAR(2)`);
   - `name` : une chaîne de caractères (`VARCHAR`);
   - `continent` : une chaîne de caractères (`VARCHAR`).

4. On considère le script SQL ci-dessous.

```sql
INSERT INTO Countries (id, code, name, continent)
    VALUES (302687, 'FR', 'France', 'EU');
INSERT INTO Countries (id, code, name, continent)
    VALUES (302684, 'ES', 'Spain', 'EU');
SELECT * FROM Countries;
```

Les deux premières instructions (appelées *requêtes* en SQL) permettent d'insérer deux **enregistrements** (lignes) dans la table.
La dernière instruction permet de récupérer (et d'afficher) la **relation** (table) `Countries`.

(a) Recopier et exécuter ce script.

(b) Récupérer sur mon site le fichier nommé countries.sql.

(c) Dans *DB Browser for SQLite*, importer ce fichier en cliquant sur *Ouvrir un fichier SQL* (petite icône au-dessus de la fenêtre SQL). Exécuter le script ainsi chargé. Que s'est-il passé ?

5. Copier la première requête du code de la question précédente et la coller dans le zone pour exécuter du SQL.

Exécuter 5 fois cette requête en cliquant 5 fois sur la petite flèche en haut.

Effacer tout ce qu'il y a dans le zone SQL puis taper :

```sql
SELECT * FROM Countries WHERE name="France";
```

Qu'observe-t-on ? Est-ce souhaitable ?

Vocabulaire

Une **base de données relationnelle** est une collection de données structurées
en (une ou plusieurs) **relations**, présentées sous la forme de **tables** à l'utilisateur.
Chaque relation modélise une classe d'objets du monde réel.

Dans une relation, les **attributs** (les colonnes) modélisent des caractéristiques de la classe d'objets.
Les valeurs associées à un objet forment un **enregistrement** (une ligne), appelé aussi n-uplet.

Chaque attribut possède un **domaine de valeurs** : c'est l'ensemble des valeurs autorisées pour cet attribut, en général présenté sous la forme d'un type (`INT`, `CHAR`, `DATE`, *etc.* – voir liste).

Définition

Un identifiant (appelé aussi clé) d'une table (relation) est la donnée d'un ou plusieurs attributs qui permet d'identifier de manière unique un enregistrement. En d'autres termes, deux enregistrements ne peuvent avoir le même identifiant.
Dans une table (relation), il y a en général plusieurs choix possibles pour un identifiant (appelés clés candidates).
La donnée d'un identifiant d'une table (relation) est appelée clé primaire.

Exercice 2

On considère une table regroupant des informations sur les élèves d'un lycée, dont les attributs sont les suivants : nom, prénom, date de naissance, adresse, numéro de sécurité sociale.

Parmi les données ci-dessous, lesquelles peuvent être utilisées comme clé primaire ? Justifier.

- nom ;
- adresse ;
- (nom, prénom) ;
- (nom, prénom, adresse) ;
- numéro de sécurité sociale ;
- (nom, numéro de sécurité sociale).

Exercice 3

Une table contient des informations sur les trajets réalisés par les utilisateurs d'un service de location de vélos. Les attributs de cette table sont les suivants : date, heures de départ et d'arrivée, stations de départ et d'arrivée. L'unicité des enregistrements est-elle garantie ? Proposer une solution afin de pouvoir définir une clé primaire.

Définition

Dans une table, une clé étrangère est un attribut qui fait référence à la clé primaire d'une autre table. Le domaine de valeurs d'une clé étrangère est donc l'ensemble des valeurs de la clé primaire référencée.

Définition

Le schéma d'une table (ou relation) est la liste de ses attributs (et de leurs domaines de valeurs), où sont précisées la clé primaire (en général par un soulignement) et les clés étrangères (en général par un astérisque).
Le **schéma** d'une base de données relationnelle consiste en la donnée des schémas de ses tables (ou relations).

Pourquoi utiliser plusieurs tables ?

Supposons qu'on souhaite constituer une base de données décrivant les cours des élèves d'une école. Il est tout à fait possible de n'utiliser qu'un seul tableau dont le début ressemblerait à ce qui suit.

eleve	classe	matiere	professeur
Harry Potter	T3	Arithmancie	Septima Vector
Harry Potter	T3	Vol de balai	Renee Bibine
Harry Potter	T3	Sortileges	Filius Flitwick
Harry Potter	T3	Dcfm*	Gilderoy Lockhart
Hermione Granger	T3	Arithmancie	Septima Vector
Hermione Granger	T3	Vol de balai	Renee Bibine
Hermione Granger	T3	Sortileges	Filius Flitwick
Hermione Granger	T3	Dcfm*	Gilderoy Lockhart
...	...	...	...

* `Dcfm` : Défense contre les forces du mal.

Mais combien de lignes devrait-on modifier si le professeur Gilderoy Lockhart était remplacé par Remus Lupin ? Et si l'élève Hermione Granger changeait de classe, avec une panoplie de professeurs différents ?

Avec une base de données relationnelle, pourvu que son schéma soit suffisamment bien conçu, chacune de ces modifications se résumerait à la mise à jour d'**un seul enregistrement**.

2 Les SGBD

Un système de gestion de bases de données (SGBD ; en anglais : *Database Management System – DBMS*) est une bibliothèque logicielle qui permet de **lire, écrire et modifier les données** d'une base de données.

Des exemples de SGBD, sous différentes licences, sont : MySQL, Oracle Database, H2, PostgreSQL, SQLite.

L'utilisateur d'un SGBD manipule les données en effectuant des **requêtes**, souvent écrites dans le langage SQL (*Structured Query Language*). **La complexité du traitement effectué par le SGBD est caché de l'utilisateur**.

De plus, les SGBD sont équipés de mécanismes garantissant :

- la **performance** d'accès aux données : à l'exécution, les requêtes sont analysées de sorte que le traitement soit le plus efficace possible (optimisations déduites du calcul relationnel) ;
- la gestion des accès **concurrents** : en cas d'accès simultanés, l'intégrité des données est préservée ;
- la **sécurisation** des accès : une autorisation (identifiant et mot de passe) est requise ;
- la **persistance** et la protection des données : le stockage est conçu de manière à éviter la perte ou la corruption de données (copies de sauvegarde, fichier journal des modifications effectuées) ;
- la **cohérence** des données : lors de modifications de la base de données, des vérifications sont effectuées pour assurer leur compatibilité avec les contraintes imposées par le schéma relationnel.

Les contraintes du schéma relationnel

Le schéma relationnel définit pour certains attributs des contraintes, appelées **contraintes d'intégrité**, qui permet de maintenir la cohérence de la base de données en cas de mise à jour. Parmi elles, on peut citer :

- la contrainte de clé primaire : un attribut (ou un ensemble d'attribut) déclaré comme **clé primaire** ne peut avoir la même valeur dans deux enregistrements différents ;
- la contrainte d'intégrité référentielle : si un attribut est déclaré comme **clé étrangère**, chaque enregistrement doit avoir pour valeur une valeur enregistrée dans l'attribut référencé par la clé étrangère ;
- la contrainte d'unicité : l'attribut ne peut avoir la même valeur dans deux enregistrements différents ;
- la contrainte d'existence : l'attribut doit avoir une valeur renseignée dans chacun des enregistrements ;
- les contraintes spécifiques : l'attribut doit vérifier une condition spécifique.

3 Le langage SQL

Afin de pouvoir utiliser des SGBD, plusieurs langages informatiques ont été élaborés. Le plus populaire est le SQL (*Structured Query Language* – langage de requêtes structurées), développé en 1986.

Parmi les éléments du langage SQL figurant explicitement au programme, on trouve :

- La **lecture** de données. Syntaxe : `SELECT attributs FROM Table WHERE condition`

  Pour chaque enregistrement dans la `Table` vérifiant la `condition`, cette instruction extrait les valeurs des `attributs` indiqués (ceux-ci étant séparés par des virgules s'il y en a plusieurs).
  Pour extraire les valeurs de tous les attributs, on utilise l'étoile (`*`).

  Exemples :

```
SELECT name, city FROM Airports WHERE country='FR';
SELECT * FROM Airports WHERE country='FR';
```

- La **jointure** de tables. Syntaxe : `R1 JOIN R2 ON R1.pk=R2.fk`

  Cette opération construit une table dont les attributs sont ceux des tables `R1` et `R2`, et dont les enregistrements sont ceux de `R1` et `R2` mis « côte à côte », tels que la condition de jointure `R1.pk=R2.fk` est réalisée.
  Très souvent, l'attribut `fk` de la table `R2` est une clé étrangère qui fait référence à la clé primaire `pk` de `R1`.

  Exemples :

```
SELECT name, num_bikes_available FROM Information
    JOIN Status ON Information.id = Status.id;
SELECT name, capacity, num_bikes_available FROM Information
    JOIN Status ON Information.id = Status.id
    WHERE capacity>50;
```

- L'**écriture** de données. Syntaxe : `INSERT INTO Table(nom_attributs) VALUES (valeurs)`

  Ajoute un enregistrement dans la `Table` des `valeurs` indiquées (séparées par des virgules) pour chacun des attributs énumérés dans l'ordre de `nom_attributs`. Les noms des attributs peuvent être omis si les valeurs sont énumérées dans l'ordre de déclaration des attributs.

  Exemples :

```
INSERT INTO Countries(id, code, name, continent)
    VALUES (302687, 'FR', 'France', 'EU');
INSERT INTO Information
    VALUES ('2515907297', 'Pasteur - Gambetta',
        48.84572797384512, 2.4236011505126953, '19');
```

(TSVP)

- La **mise à jour** de données. Syntaxe : `UPDATE Table SET attribut = valeur WHERE condition`.

  Dans tous les enregistrements de la `Table` qui vérifient la `condition`, cette instruction affecte une nouvelle `valeur` à l'`attribut` indiqué.

  Exemples :

```
UPDATE Status SET num_bikes_available = 0 WHERE id='2515907297';
UPDATE Status
    SET num_bikes_available = num_bikes_available+1
    WHERE id='2515907297';
```

- La **suppression** de données. Syntaxe : `DELETE FROM Table WHERE condition`.

  Cette instruction supprime tous les enregistrements de la `Table` vérifiant la `condition`.

  Exemples :

```
DELETE FROM Countries WHERE name='France';
```

→ Le programme fait également référence aux éléments suivants :

- le mot-clé `DISTINCT`, qui permet de **supprimer les doublons** dans une requête qui en contiendraient. Exemple :

```
SELECT DISTINCT country FROM Airports;
```

- la clause `ORDER BY`, qui permet de **trier** une lecture selon un attribut (ou plusieurs). Exemples :

```
SELECT name, num_bikes_available FROM Information
    JOIN Status ON Information.id = Status.id
    ORDER BY name;
SELECT name, num_bikes_available FROM Information
    JOIN Status ON Information.id = Status.id
    ORDER BY num_bikes_available, name;
```

- les **fonctions d'agrégation**, qui réalisent une opération arithmétique sur les enregistrements obtenus lors d'une lecture. Exemples :

```
SELECT COUNT(name) FROM Information;          -- nombre de stations
SELECT MAX(altitude) FROM Airports;           -- altitude maximale
SELECT AVG(altitude) FROM Airports;           -- altitude moyenne (average)
SELECT SUM(num_bikes_available) FROM Status;  -- nombre de vélos disponibles
```

Quelques remarques en vrac concernant le langage SQL :

- une instruction peut être écrite sur plusieurs lignes, mais doit se terminer par un point-virgule ;
- le test d'égalité est le symbole = (non doublé) ;
- le nom d'un attribut peut être préfixé par le nom de la table dont il dépend (`Table.attribut`) ;
- les chaînes de caractères sont entourées de guillemets droits simples ;
- l'échappement d'un guillemet s'effectue en le doublant ;
- la convention générale est d'écrire les mots-clés en majuscules, la première lettre des tables en majuscule, les attributs en minuscules ;

Exercice 4

On considère une base de données qui décrit des informations sur des articles et des clients qui achètent ces articles. Le schéma relationnel de cette base est le suivant, où la clé primaire de chaque table est soulignée et les attributs qui forment une clé étrangère sont suivis de `*`.

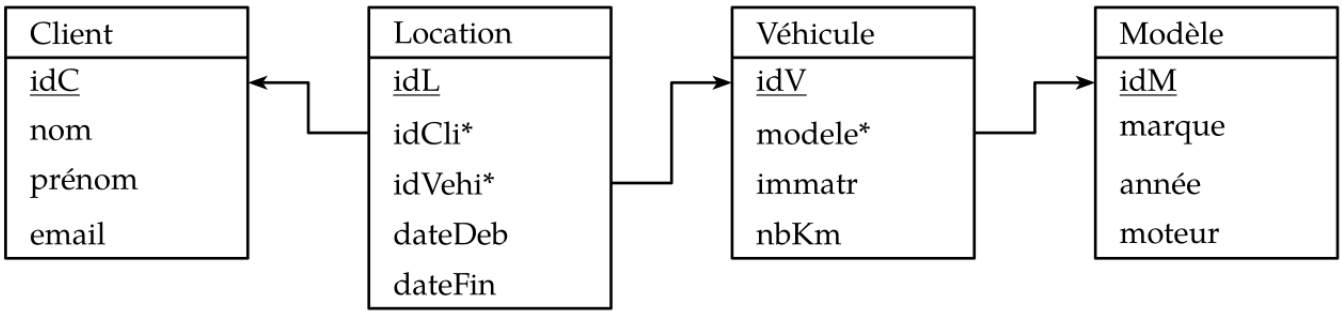
- Achats (aid*, cid*, dateachat, quantité)
- Articles (aid, rayon, prix)
- Clients (cid, nom, datenaiss)

1. Est-il possible pour un client d'acheter un même article plusieurs fois ?
2. Quelles sont les instructions SQL permettant de créer les tables ci-dessus ? (On peut s'aider du logiciel utilisé)
3. On dispose de trois fichiers `achats.sql`, `articles.sql` et `clients.sql` qui permettent d'effectuer des enregistrements dans chacune des tables ci-dessus.

   On exécute les instructions SQL contenues dans ces fichiers : quel ordre chronologique peut-on choisir pour alimenter la base de données avec les fichiers mis à disposition ?

Exercice 5

Afin de gérer les réservations de ses véhicules, une société de location de voiture dispose d'une base de données dont le schéma relationnel est le suivant :



On suppose que ces tables ont été définies dans un SGBD basé sur le langage SQL.

On admet de plus que les attributs `dateDeb` et `dateFin` de la table `Location` sont au format `DATETIME`. Une date et heure au format `DATETIME` s'écrit à l'aide de la chaîne de caractères '`AAAA-MM-JJ HH:MM:SS`'.

On admet aussi que l'attribut `moteur` de la table `Modèle` contient une chaîne de caractères pouvant valoir soit `essence`, soit `électrique`.

Il est possible de comparer deux valeurs au format `DATETIME` à l'aide des opérateurs usuels (<, >=, *etc.*).

1. Dans la table `Client`, indiquer, en justifiant, un autre choix possible pour la clé primaire.
2. À quelle condition est-il possible de supprimer un enregistrement dans la table `Véhicule` ?
3. Mme Dupont a loué une voiture le 1er février 2022 à 8h. L'instruction SQL suivante a permis l'enregistrement de cette location dans la base de données.

```sql
INSERT INTO Location VALUES (9268, 1345, 245, '2022-02-01 08:00:00', null);
```

Mme Dupont a restitué le véhicule le 3 février 2022 à 16h : quelle requête SQL permet de mettre à jour en conséquence la base de données ?

4. Compléter la requête SQL ci-dessous de manière à lister les immatriculations des véhicules de la société qui ont plus de 50 000 km.

```sql
SELECT ??? FROM Véhicule WHERE ???;
```

5. Écrire une requête SQL qui permet de connaître le nombre de locations en cours.
6. Écrire une requête SQL qui affiche le nombre de véhicules électriques que la société possède.
7. Quelle requête SQL permet d'extraire les adresses électroniques des clients qui ont réservé un véhicule électrique durant le mois de janvier 2022 ?