DS3

Exercice 1

(d'après Centrale PSI 2016)

Afin d'éviter les collisions entre avions, les altitudes de vol en croisière sont normalisées. Dans la majorité des pays, les avions volent à une altitude multiple de 1000 pieds (un pied vaut 30,48 cm) au-dessus de la surface isobare à 1013,25 hPa. L'espace aérien est ainsi découpé en tranches horizontales appelées niveaux de vol et désignées par les lettres « FL » (*flight level*) suivies de l'altitude en centaines de pieds : « FL310 » désigne une altitude de croisière de 31000 pieds au-dessus de la surface isobare de référence.

Eurocontrol est l'organisation européenne chargée de la navigation aérienne, elle gère plusieurs dizaines de milliers de vol par jour. Toute compagnie qui souhaite faire traverser le ciel européen à un de ses avions doit soumettre à cet organisme un plan de vol comprenant un certain nombre d'informations : trajet, heure de départ, niveau de vol souhaité, etc. Muni de ces informations, Eurocontrol peut prévoir les secteurs aériens qui vont être surchargés et prendre des mesures en conséquence pour les désengorger : retard au décollage, modification de la route à suivre, etc.

Nous modélisons (de manière très simplifiée) les plans de vol gérés par Eurocontrol sous la forme d'une base de données comportant trois tables :

- la table `vol` qui répertorie les plans de vol déposés par les compagnies aériennes ; elle contient les colonnes
  - `id_vol` : numéro du vol (chaine de caractères) ;
  - `comp` : le code de la compagnie aérienne qui opère le vol (chaine de caractères) ;
  - `depart` : code de l'aéroport de départ (chaine de caractères) ;
  - `arrivee` : code de l'aéroport d'arrivée (chaine de caractères) ;
  - `jour` : jour du vol (de type date, affiché au format `aaaa-mm-jj`) ;
  - `heure` : heure de décollage souhaitée (de type time, affiché au format `hh:mi`) ;
  - `niveau` : niveau de vol souhaité (entier).

id_vol	comp	depart	arrivee	jour	heure	niveau
AF1204	AF	CDG	FCO	2016-05-02	07 :35	300
AF1205	AF	FCO	CDG	2016-05-02	10 :25	300
AF1504	AF	CDG	FCO	2016-05-02	10 :05	310
AF1505	AF	FCO	CDG	2016-05-02	13 :00	310

Figure 1 Extrait de la table `vol`

- la table `aeroport` qui répertorie les aéroports européens ; elle contient les colonnes
  - `id_aero` : code de l'aéroport (chaine de caractères) ;
  - `ville` : principale ville desservie (chaine de caractères) ;
  - `pays` : pays dans lequel se situe l'aéroport (chaine de caractères).

id_aero	ville	pays
CDG	Paris	France
ORY	Paris	France
MRS	Marseille	France
FCO	Rome	Italie

Figure 2 Extrait de la table `aeroport`

- la table `compagnie` qui répertorie les compagnies aériennes ; elle contient les colonnes
  - `id_comp` : code de la compagnie (chaine de caractères) ;
  - `nom` : le nom de la compagnie (chaine de caractères) ;
  - `pays` : pays dans lequel la compagnie est domiciliée (chaine de caractères).

id_comp	nom	pays
AF	Air France	France
IB	Iberia	Espagne

Figure 3 Extrait de la table `compagnie`

Les types SQL `date` et `time` permettent de mémoriser respectivement un jour du calendrier grégorien et une heure du jour. Deux valeurs de type `date` ou de type `time` peuvent être comparées avec les opérateurs habituels(=, <, <=, etc.). La comparaison s'effectue suivant l'ordre chronologique. Ces valeurs peuvent également être comparées à une chaine de caractères correspondant à leur représentation externe (`'aaaa-mm-jj'` ou `'hh:mi'`).
Pour les requêtes qui suivent, on n'utilisera ni LIMIT, ni OFFSET.

1. Écrire une requête SQL qui fournit toutes les données des vols au départ de l'aéroport Charles de Gaulle (code `CDG`) le 2 mai 2016.
2. Écrire une requête SQL qui fournit le nombre de vols qui doivent décoller dans la journée du 2 mai 2016 avant midi (*au sens large*).

3. Écrire une requête SQL qui fournit la liste des numéros de vols qui atterrissent à un des aéroports de Paris le 2 mai 2016.
4. Écrire une requête SQL qui fournit la liste des noms des compagnies domiciliées en France et qui ont opéré au moins un vol au départ d'un aéroport italien le 2 mai 2016.
5. Écrire une requête SQL qui fournit la liste des codes des compagnies et le niveau de vol moyen qu'elles utilisent.
6. Écrire une requête SQL qui fournit la liste des codes des compagnies (sans répétition) qui ont opéré au moins 10 vols au départ de l'aéroport Charles de Gaulle (code `CDG`) le 2 mais 2016.
7. Écrire une requête SQL qui fournit la valeur du niveau minimal utilisé par un avion le 2 mai 2016.
8. Écrire une requête SQL qui fournit la liste (sans répétition) des noms des compagnies ayant utilisé le niveau minimal le 2 mai 2016.
9. Que fait la requête suivante ?

```
SELECT id_vol
FROM vol
     JOIN aeroport AS d ON d.id_aero = depart
     JOIN aeroport AS a ON a.id_aero = arrivee
WHERE
     d.pays = 'France' AND
     a.pays = 'France' AND
     jour = '2016-05-02'
```

10. Certains vols peuvent engendrer des conflits potentiels : c'est par exemple le cas lorsque deux avions suivent un même trajet, en sens inverse, le même jour et à un même niveau. Écrire une requête SQL qui fournit la liste des couples (Id_1 , Id_2) des identifiants des vols dans cette situation le 26 mai 2016.

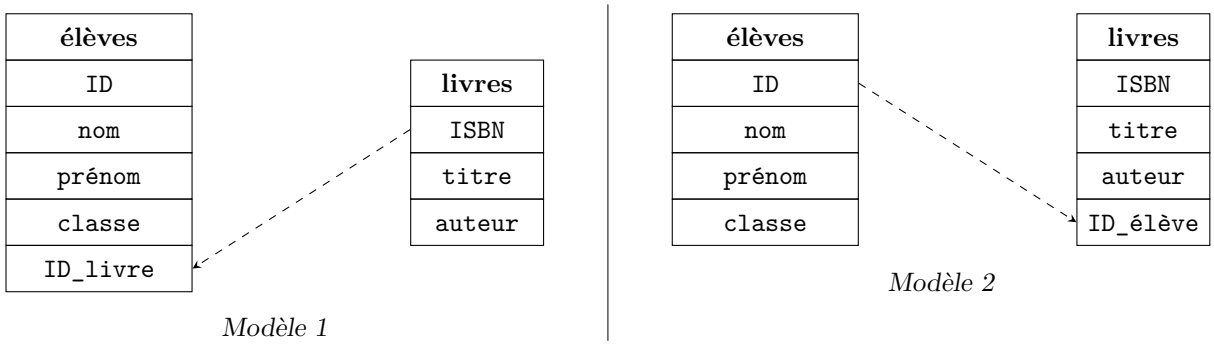
Exercice 2

La bibliothèque d'une école souhaite gérer ses prêts de livres à l'aide d'une base de données. Pour cela elle considère deux entités :

— Les `élèves` dont les caractéristiques sont : `ID`, `nom`, `prénom` et `classe`.
— Les `livres` dont les caractéristiques sont : `ISBN`, `titre` et `auteur`.

L'attribut `ID` est un identifiant unique pour chaque élève et l'`ISBN` est un code unique propre à chaque livre (mais qui est le même pour d'éventuels exemplaires différents du même livre).
Chaque élève a la possibilité d'emprunter jusqu'à 4 livres en même temps.

1. On suppose dans un premier temps que la bibliothèque ne dispose que d'un seul exemplaire de chaque livre.
   a) Quel est le type (cardinalité) de l'association `livres - élèves` ?
   b) Afin de faire correspondre les deux entités, on doit rajouter un attribut à une des deux tables ; on a donc deux modèles possibles :



   Quel modèle est le plus adapté pour éviter les répétitions inutiles de données ? (Justifier)

2. On suppose cette fois que la bibliothèque dispose de plusieurs exemplaires de chaque livre. On s'intéresse seulement aux titres des livres empruntés par les élèves (et non à l'exemplaire précis du livre emprunté), il est donc inutile de rajouter un identifiant supplémentaire dans la table `livres` pour repérer chaque exemplaire.
   Quel problème pose le modèle précédemment choisi ?
   Comment peut-on modéliser la situation de façon efficace ? Vous pouvez modifier les tables `élèves` et `livres` et en introduire d'autres (en expliquant vos choix) et en indiquant pour chacune des tables quelles sont leurs éventuelles clés primaires (à mettre en majuscule) et clés étrangères. Pour décrire le modèle, vous utiliserez une représentation des tables similaire à celle utilisée dans les modèles 1 et 2 précédents.
3. Comment adapter le modèle précédent si on souhaite connaître précisément les exemplaires des livres empruntés par les élèves ?