TD #23 — Trois exercices sur les bases de données (avec solutions)

Exercice 1. *XENS Info. B 2016*

Une représentation simplifiée, réduite à deux tables, de la base de données d'un réseau social est donnée par le schéma :

— **individus** : (id : entier ; nom : texte ; prenom : texte)
— **liens** : (id1 : entier ; id2 : entier)

La table **individus** répertorie les individus, et la table **liens** répertorie les liens d'amitiés entre individus.
On supposera par ailleurs que pour tout couple `(x,y)` dans la table **liens**, le couple `(y,x)` est également présent dans la table.

1. Écrire une requête SQL qui renvoie les identifiants des amis de l'individu d'identifiant `x`.

   On propose :
   ```
   SELECT id1 FROM LIENS WHERE id2 = x;
   ```

2. Écrire une requête SQL qui renvoie les (noms,prénoms) des amis de l'individu d'identifiant `x`.

   On propose :
   ```
   SELECT nom, prenoms FROM INDIVIDUS
   JOIN LIENS ON id = id1
   WHERE id2 = x;
   ```

3. Écrire une requête SQL qui renvoie les identifiants des individus qui sont amis avec au moins un ami de l'individu d'identifiant `x`.

   On propose :
   ```
   SELECT DISTINCT id1 FROM LIENS l1
   JOIN LIENS l2 ON l1.id2 = l2.id1 WHERE l2.id2 = x;
   ```
   On peut visualiser l'idée derrière cette requête en représentant la table qu'elle construit :

l1.id1	l1.id2	l2.id1	l2.id2
a	b	b	x

Exercice 2. *Sur les routes*

On considère une base de données recensant des villes ainsi que des routes reliant des villes entre elles. Elle est organisée autour de deux tables **villes** et **routes** de schémas suivant :

— **villes** : (id : entier ; nom : texte ; population : entier)
— **routes** : (id : texte ; id_ville1 : entier ; id_ville2 entier ; longueur : entier)

Les attributs *id_ville1* et *id_ville2* de la table **routes** sont chacune une clef étrangère faisant référence à l'attribut *id* de la table **villes**.
On précise également que toute route, identifiée de façon unique par son *id*, étant systématiquement à double-sens, elle ne doit apparaître qu'une seule fois dans la table **routes**. Par ailleurs, deux villes peuvent être reliées par plusieurs routes distinctes.
On donne un exemple de données pour chacune de ces tables dans la Figure 1.

1. Expliquez quel problème pourrait survenir si l'on supprimait l'attribut *id* de la table **routes**.

   Sans attribut *id*, une même route pourrait être présente deux fois dans la table avec *id_ville1* et *id_ville2* inversés, ce qui ne respecterait pas les spécifications de **routes**. Il serait aussi impossible de représenter (plus de) deux routes distinctes de même longueur entre les deux mêmes villes.

Les questions suivantes demandent d'écrire des requêtes SQL pour les tables **routes** et **villes** de schémas définis ci-dessus. Attention : vos requêtes doivent fonctionner pour toutes valeurs possibles des attributs ; il n'est par exemple pas possible de supposer *a priori* que l'attribut *id* associé à la ville d'Allevard est 2.

villes		
id	nom	population
1	Grenoble	156 389
2	Allevard	3 862
3	Domène	6 777
4	Aussois	682

routes			
id	id_ville1	id_ville2	longueur
D1090	1	2	40
D523	3	1	11
D11	3	2	34
D525	2	4	123
D523-2	3	1	13

FIGURE 1 – Exemple de tables **villes** et **routes**

2. Écrivez une requête SQL renvoyant les noms de toutes les villes.

```sql
SELECT nom FROM villes;
```

3. Écrivez une requête SQL renvoyant la population de la ou les villes les moins peuplées.

```sql
SELECT MIN(population) FROM villes;
```

4. Écrivez une requête SQL renvoyant les identifiants des routes ainsi que leurs longueurs, dans l'ordre de longueur décroissant.

```sql
SELECT id, longueur FROM routes ORDER BY longueur DESC;
```

5. Écrivez une requête SQL renvoyant sans doublons les noms de toutes les villes voisines de Grenoble (c'est à dire telles qu'il existe une route entre cette ville et Grenoble).

```sql
SELECT v2.nom FROM villes v1
  JOIN routes r ON r.id_ville1 = v1.id
  JOIN villes v2 ON r.id_ville2 = v2.id
  WHERE v1.nom = 'Grenoble'
UNION
SELECT v2.nom FROM villes v1
  JOIN routes r ON r.id_ville2 = v1.id
  JOIN villes v2 ON r.id_ville1 = v2.id
  WHERE v1.nom = 'Grenoble';
```

Exercice 3. *Photographie aérienne*

Dans cet exercice, on étudie une base de données de photographies aériennes utilisées en cartographie. Les informations que l'on souhaite pouvoir manipuler à travers cette base sont les zones photographiées (découpées en *tuiles*), les aéronefs utilisés, les organismes de cartographie commanditant ou réalisant les prises de vues, et les photos elles-mêmes.
La base de données est constituée des tables de schémas suivants :

— **tuile** : (id : entier ; latN : flottant ; longE : flottant ; long : flottant ; larg : flottant)
— **org** : (id : entier ; nom : texte)
— **aeronef** : (id : entier ; nom : texte)
— **photo** : (id : entier ; id_tuile : entier ; id_aeronef : entier ; id_org : entier ; annee : entier)

On donne ci-dessous un exemple de données pour chacune de ces tables. Attention : les informations contenues dans ces exemples sont données à titre indicatif : vous ne pouvez pas les utiliser pour répondre aux questions !

tuile				
id	latN	longE	long	larg
1	45.187	5.725	2000	2000
2	45.415	6.184	2000	2000

org	
id	nom
1	Institut Géographique National
2	Ordnance Survey

aeronef	
id	nom
1	Beechcraft Super King Air 200 T
2	Boeing B-17
3	Hurel-Dubois HD-34

photo				
id	id_tuile	id_aeronef	id_org	annee
1	1	1	1	2025
2	1	2	1	1979
3	2	3	1	1984
4	2	1	1	2025
5	1	2	2	1979

Étude de la base de données

1. Donnez (en justifiant) le type de l'association « a été prise par » entre une photo (aérienne) et un aéronef.

> Une photo a forcément été prise par un aéronef, et un aéronef a pu prendre plusieurs photos : l'association est donc de type $1 - *$.

2. Les informations à propos des tables ci-dessus n'identifient pas les éventuelles clefs étrangères qui y sont présentes. Donnez une clef étrangère de la table **photo** et la clef candidate qu'elle référence.

> Par la réponse à la question précédente, l'attribut *id_aeronef* de la table **photo** doit logiquement être une clef étrangère pour la table **aeronef**, y référençant l'attribut *id*.

On aimerait enrichir la base de données pour y stocker l'information des flottes d'aéronefs utilisés par les différents organismes : chaque organisme utilise un ou plusieurs aéronefs d'un ou plusieurs type, et chaque type d'aéronef peut être utilisé par un ou plusieurs organismes.

3. Proposez une modification de la base de données (qui y ajoute une ou plusieurs tables, et/ou modifie une ou plusieurs tables existantes) permettant de stocker cette information. Cette modification devra entre autres permettre de répondre à des questions comme « quels sont les noms d'aéronefs utilisés par … » ; « combien d'aéronefs … sont utilisés par … ».

> Une solution consisterait à ajouter une table **flotte** d'attributs entiers *id_org*, *id_aeronef* et *effectif*, où les deux premiers attributs référenceraient les attributs *id* de respectivement **org** et **aeronef**, et le dernier donnant le nombre d'aéronefs d'identifiant indiqué par le second attribut possédé par l'organisme d'identifiant indiqué par le premier.

Interrogation de la base de données

4. Écrivez une requête SQL qui fournit toutes les informations à propos des photos prises en 2025.

> On propose :
> ```
> SELECT * FROM photo WHERE annee = 2025;
> ```

5. Écrivez une requête SQL qui fournit toutes les informations à propos des photos prises par l'organisme s'appelant *Institut Géographique National*

On propose :

```sql
SELECT * FROM photo
JOIN org ON id_org = org.id
WHERE org.nom = 'Institut Géographique National';
```

6. Écrivez une requête SQL qui fournit toutes les informations à propos des photos prises l'année la plus ancienne (1979 pour les tables données en exemple).

On propose :

```sql
SELECT * FROM photo WHERE annee = (SELECT MIN(annee) FROM photo);
```

7. Écrivez une requête SQL qui fournit pour chaque aéronef le nombre total de photos qu'il a prises. La réponse à la requête devra être constituée de deux colonnes : l'une donnant le nom des aéronefs, et la seconde nommée `nb_photos` donnant le nombre de photos.

On propose :

```sql
SELECT aeronef.nom, COUNT(*) as nb_photos FROM photo
JOIN aeronef ON id_aeronef = aeronef.id
GROUP BY aeronef.id;
```

8. Écrivez une requête SQL qui fournit pour chaque organisme le nombre de photos prises par année (où il a pris des photos). La réponse à la requête devra être constituée de trois colonnes : l'une donnant le nom d'un organisme, la seconde donnant une année, et la troisième nommée `nb_photos` le nombre de photos prises par l'organisme cette année là.

On propose :

```sql
SELECT org.nom, photo.annee, COUNT(*) nb_photos FROM photo
JOIN org on id_org = org.id
GROUP BY org.nom, photo.annee;
```

9. Écrivez une requête SQL qui fournit pour chaque organisme le nombre moyen de photos qu'il prend par année, où cette moyenne est calculée en prenant seulement en compte les années d'activité (où des photos ont été prises). La réponse à la requête devra être constituée de deux colonnes : l'une donnant le nom d'un organisme et la seconde la moyenne calculée.

On propose :

```sql
SELECT nom, AVG(nb_photos) FROM
(SELECT org.nom, photo.annee, COUNT(*) nb_photos FROM photo
JOIN org on id_org = org.id
GROUP BY org.nom, photo.annee)
GROUP BY nom;
```