Module 5 - TD Correction: les différentes opérations de l'algèbre relationnelle en pratique: opérateurs, jointures et fonctions de groupes

Paul Schuhmacher

version: 1.0

Préparation du schéma et des données

- **Créer** la base de données `mod5jointures` sur laquelle votre utilisateur `dev` a tous les droits.
- **Créer** les relations `Compagnie` et `Pilote`
- **Insérer** les données préparées

```
USE mod5jointures;

CREATE TABLE Compagnie
(comp VARCHAR(4), nrue SMALLINT(3), rue VARCHAR(20), ville VARCHAR(15), nomComp
        VARCHAR(15),
 CONSTRAINT pk_Compagnie PRIMARY KEY(comp));

CREATE TABLE Pilote
(brevet VARCHAR(6), nom CHAR(20), nbHVol DECIMAL(7,2), compa VARCHAR(4),
        chefPil VARCHAR(6),
CONSTRAINT pk_Pilote PRIMARY KEY(brevet),
CONSTRAINT fk_Pil_compa_Comp FOREIGN KEY(compa) REFERENCES Compagnie(comp),
 CONSTRAINT fk_Pil_chefPil_Pil FOREIGN KEY(chefPil) REFERENCES Pilote(brevet));

INSERT INTO Compagnie   VALUES ('AF', 124, 'Port Royal', 'Paris', 'Air
        France');
INSERT INTO Compagnie   VALUES ('SING', 7, 'Camparols', 'Singapour', 'Singapore
        AL');
INSERT INTO Compagnie   VALUES ('CAST', 1, 'G. Brassens', 'Blagnac', 'Castanet
        AL');

INSERT INTO Pilote VALUES ('PL-4', 'Henri Alquié', 3400, 'AF',NULL);
INSERT INTO Pilote VALUES ('PL-1', 'Pierre Lamothe', 450, 'AF', 'PL-4');
INSERT INTO Pilote VALUES ('PL-2', 'Didier Linxe', 900, 'AF','PL-4');
INSERT INTO Pilote VALUES ('PL-3', 'Christian Soutou', 1000, 'SING',NULL);
```

Jointures internes ([`INNER JOIN| JOIN`])

Équijointure

1. **Dessiner** le *schéma conceptuel* (formalisme UML) à partir des requêtes SQL
2. **Identifier** les clefs primaires et étrangères.

Écrire des requêtes pour extraire les données suivantes:

3. L'identité des pilotes de la compagnie de nom Air France ayant plus de 500 heures de vol (requête R1)

```
SELECT * FROM Pilote JOIN Compagnie ON Pilote.compa = Compagnie.comp WHERE
        nomComp = 'Air France' AND Pilote.nbHVol > 500;
```

4. Les coordonnées des compagnies qui embauchent des pilotes de moins de 500 heures de vol (requête R2)

```
SELECT nomComp, nrue, rue, ville FROM Pilote JOIN Compagnie ON Pilote.compa =
        Compagnie.comp WHERE Pilote.nbHVol < 500;
```

Autojointure

Cas particulier de l'*équijointure*, l'autojointure (*self-join*) relie une table *à elle même* (pratique dans le cas d'associations réflexives).

Nous allons extraire les données suivantes:

5. L'identité des pilotes placés sous la responsabilité de `'Alquié'` (R3)

```
-- Pour différencier les deux relations dans la jointure il faut les aliaser
        avec FROM NomTable [AS] AliasTable
SELECT p1.brevet, p1.nom FROM Pilote AS p1 JOIN Pilote AS p2 ON p1.chefPil =
        p2.brevet WHERE p2.nom LIKE '%Alquié%';
-- ou sans le AS (optionnel)
SELECT p1.brevet, p1.nom FROM Pilote p1 JOIN Pilote p2 ON p1.chefPil =
        p2.brevet WHERE p2.nom LIKE '%Alquié%';
```

6. La somme des heures de vol des pilotes placés sous la responsabilité des chefs pilotes de la compagnie de nom 'Air France' (R4)

   Plus difficile, plusieurs réponses possibles. Découper la requête en sous requêtes.

On découpe la requête en plusieurs petites requêtes: déjà, il faut trouver les pilotes qui ont un chef à "Air France". Il faut commencer par trouver les pilotes qui ont un chef et récupérer le chef:

```
-- Requête intermédiaire (pilotes qui ont un chef)
SELECT * FROM Pilote p1 JOIN Pilote p2 ON p1.chefPil = p2.brevet;
-- Requête intermédiaire (pilotes qui ont un chef chez "Air France")
```

```
SELECT * FROM Pilote p1     JOIN Pilote p2     ON p1.chefPil = p2.brevet
        JOIN Compagnie         ON p1.compa = Compagnie.comp
        WHERE nomComp = 'Air France';
-- Il ne reste plus qu'à faire la somme des heures de vol
SELECT SUM(p1.nbHVol) FROM Pilote p1 JOIN Pilote p2 ON p1.chefPil = p2.brevet
        JOIN Compagnie ON p1.compa = Compagnie.comp WHERE nomComp = 'Air
        France';
```

Inéquijointure

7. Les pilotes ayant plus d'expérience (i.e d'heures de vol) que le pilote de numéro de brevet 'PL-2'. (R5)

```
SELECT p1.brevet, p1.nom, p1.nbHVol, p2.nbHvol "Reference" FROM Pilote p1 JOIN
        Pilote p2 on p1.nbHVol > p2.nbHVol WHERE p2.brevet = 'PL-2';
```

8. Le titre de qualification des pilotes en raisonnant sur la comparaison des heures de vol avec un ensemble de référence, ici la table `HeuresVol` (R6). Dans notre exemple, il s'agit par exemple de retrouver le fait que le premier pilote est débutant. La table d'heures de référence est définie par la relation suivante: `HeuresVol: **titre**, basnbHVol (borne min), hautnbHVol (borne max)` avec les valeurs suivantes:

- `Débutant`: entre 0 et 500 heures
- `Initié`: entre 501 et 1000 heures
- `Expert`: entre 1001 et 5000 heures

```
--Créer la table et insérer les données
CREATE TABLE HeuresVol(titre CHAR(20), basnbHVol DECIMAL(7,2), hautnbHVol
        DECIMAL(7,2));
INSERT INTO HeuresVol VALUES ('Débutant', 0, 500);
INSERT INTO HeuresVol VALUES ('Initié', 501, 1000);
INSERT INTO HeuresVol VALUES ('Expert', 1001, 5000);

-- solution 1

SELECT h.titre, p.nom, p.nbHVol FROM Pilote p JOIN HeuresVol h ON p.nbHVol >
        h.basnbHVol AND p.nbHVol <= h.hautnbHVol;

-- solution 2 avec BETWEEN

SELECT  h.titre, p.nom, p.nbHVol
    FROM Pilote p INNER JOIN HeuresVol h ON p.nbHVol BETWEEN h.basnbHVol AND
        h.hautnbHVol;
```

Jointures externes (`[LEFT|RIGHT] OUTER JOIN`)

9. La liste des compagnies aériennes et leurs pilotes (nom compagnie, nom pilote) *même* les compagnies n'ayant pas de pilote (R7)

Vous ne pouvez pas réaliser cette requête sans une jointure externe. Quand vous identifiez le mot "même", ça sent la jointure externe. Car cela veut dire faire remonter des enregistrements qui ne satisfont pas une condition donnée (le prédicat de la clause ON)

```
-- la table à gauche (Compagnie) est dominante
SELECT nomComp, nom FROM Compagnie LEFT OUTER JOIN Pilote ON compa=comp;

-- équivalent à (symétrie). La table à droite (Compagnie toujours) est
        dominante
SELECT nomComp, nom FROM Pilote RIGHT OUTER JOIN Compagnie ON compa=comp;
```

10. La liste des pilotes et leurs qualifications (le *type d'avion* qu'un pilote peut piloter en fonction de son *brevet*), *même* les pilotes n'ayant pas encore de qualification (R8). Ces données vous sont fournies par le script SQL suivant:

```
-- Créer la table de Qualifications et insérer les données
CREATE TABLE Qualifs
    (brevet VARCHAR(6), typeAv CHAR(4), validite DATE);
INSERT INTO Qualifs VALUES ('PL-4', 'A320','2005-06-24');
INSERT INTO Qualifs VALUES ('PL-4', 'A340','2005-06-24');
INSERT INTO Qualifs VALUES ('PL-2', 'A320','2006-04-04');
INSERT INTO Qualifs VALUES ('PL-3', 'A330','2006-05-13');
```

```
-- En utilisant un LEFT OUTER JOIN
SELECT p.nom, p.brevet, q.typeAv FROM Pilote p LEFT OUTER JOIN Qualifs q ON
        p.brevet=q.brevet;
```

Opérations ensemblistes: Union (`UNION`), intersection (`INTERSECT`) et différence (`NOT IN`)

11. Les types d'avion que les deux compagnies exploitent en commun.

Créer les tables suivantes contenant les avions, et **insérer** les données.

```
-- Création de deux tables pour les avions des compagnies
CREATE TABLE AvionsdeAF
(immat CHAR(6), typeAvion CHAR(10), nbHVol DECIMAL(10,2),
CONSTRAINT pk_AvionsdeAF PRIMARY KEY (immat));

CREATE TABLE AvionsdeSING
(immatriculation CHAR(6), typeAv CHAR(10), prixAchat DECIMAL(14,2),
CONSTRAINT pk_AvionsdeSING PRIMARY KEY (immatriculation));

-- Insertion des données
INSERT INTO AvionsdeAF VALUES ('F-WTSS', 'Concorde', 6570);
INSERT INTO AvionsdeAF VALUES ('F-GLFS', 'A320', 3500);
INSERT INTO AvionsdeAF VALUES ('F-GTMP', 'A340', NULL);
```

```
INSERT INTO AvionsdeSING VALUES ('S-ANSI', 'A320', 104500);
INSERT INTO AvionsdeSING VALUES ('S-AVEZ', 'A320', 156000);
INSERT INTO AvionsdeSING VALUES ('S-MILE', 'A330', 198000);
INSERT INTO AvionsdeSING VALUES ('F-GTMP', 'A340', 204500);
```

```
SELECT typeAvion FROM AvionsdeAF INTERSECT SELECT typeAv FROM  AvionsdeSING;
```

12. Les appareils utilisés par les deux compagnies.

```
SELECT immat FROM  AvionsdeAF INTERSECT SELECT immatriculation FROM
        AvionsdeSING;
```

13. *Tous* les types d'avions que les deux compagnies exploitent.

```
SELECT typeAvion FROM AvionsdeAF UNION Select typeAV FROM AvionsdeSING;
```

14. La même que la requête précédente, mais **avec les duplicatas**.

```
SELECT typeAvion FROM AvionsdeAF UNION ALL Select typeAV FROM AvionsdeSING;
```

15. Les types d'avions exploités par la compagnie `'SING'` mais pas par la compagnie `'AF'` ?

```
-- avec sous-requête
SELECT typeAV FROM AvionsdeSING WHERE typeAv NOT IN (SELECT typeAvion FROM
        AvionsdeAF);
```

Opération de regroupement et fonctions de groupes (*aggregate functions*)

A l'aide des clauses `ORDER BY`, `GROUP BY` et `HAVING`, **écrire** les requêtes qui retournent les résultats suivants.

16. La somme d'heures de vol par compagnie, ordonné du plus petit au plus grand (utiliser la fonction de groupe `SUM`)

```
SELECT compa,SUM(nbHvol) 'SommeHVol'
FROM Pilote
GROUP BY compa
ORDER BY SommeHVol;
```

17. Le nombre moyen et l'écart type du nombre d'heures de vol par compagnie (utiliser les fonctions de groupe `AVG` et `STDDEV`)

```
SELECT compa,AVG(nbHvol), STDEV(nbHVol)
FROM Pilote
GROUP BY compa;
```

18. Le nombre d'heures de vol le plus élevé pour chaque compagnie (utiliser la fonction `MAX`).

```sql
SELECT compa,MAX(nbHvol)
FROM Pilote
GROUP BY compa;
```

19. Les compagnies et le nombre de pilotes des compagnies ayant plus d'un pilote (utiliser `HAVING`)

```sql
SELECT compa,COUNT(brevet)
FROM Pilote
GROUP BY compa
HAVING COUNT(brevet) >=2;
```

> *Pour en savoir plus sur les clauses et les fonctions de regroupements, lisez cette partie de la documentation. Voir la liste des fonctions de groupe.*

Références

Aller plus loin sur les sous-requêtes:

- Common Table Expressions (CTE), permettent d'aliaser des requêtes pour les utiliser comme sous requête. S'écrit avec la clause WITH. **Les CTE sont à privilégier par rapport aux sous-requêtes**: plus lisibles, peuvent être réutilisées.
- L'implémentation de `GROUP BY` en MySQL