Examen

- Date : **28 Février 2019**
- Beaucoup de questions, il faut en réussir au minimum une dizaine pour avoir 50%

Rappels SQL

Travail sur Base de données *Bibliotheque*

Représentation Base de Données

- BDD composée de **Tables**
- Tables composées d'**Attributs** (types: **int**, **varchar**, **date**, **text**, **enum**, etc)
- id **(PK)** numéro unique pour définir un enregistrement
- xid **(FK)** numéro correspondant à un id d'une autre table *(Esclave marqué par le maître)*

Révisions memento Bruno

Sélection simple

```
SELECT personnes.PerNom FROM personnes
```

```
SELECT * FROM personnes
```

Note : * est un raccourci qui permet de selectionner tous les attributs.

Sélection avec filtre

```
SELECT personnes.PerNom FROM personnes
WHERE personnes.PerPrenom = "Christophe"
```

Sélection Avec tri

```
SELECT personnes.PerNom, personnes.PerPrenom
FROM personnes
ORDER BY personnes.PerNom DESC, personnes.PerPrenom ASC
```

Notes :

- Selection de plusieurs attributs ici grâce à la virgule.
- Ne rien préciser lors du **ORDER BY** équivaut à ORDER BY ASC.

Sélection sans doublon

```
SELECT DISTINCT personnes.PerPrenom FROM personnes
WHERE personnes.PerPrenom = "Christophe"
```

Selection avec jointure

```
SELECT auteurs.AutPrenom, auteurs.AutNom, personnes.PerPrenom, personnes.PerNom FROM auteurs, personnes
WHERE auteurs.AutPrenom = personnes.PerPrenom
```

```
SELECT personnes.PerNom FROM personnes
WHERE personnes.PerPrenom LIKE "_hristo%"
```

Note : **LIKE** à utiliser pour détecter un pattern dans une colonne (_ => 1 Char, % => plusieurs Char).

Sélection avec requête imbriquée

```
SELECT * FROM auteurs
WHERE auteurs.AutDateNaiss > (
    SELECT personnes.PerDateNaiss FROM personnes
    WHERE personnes.PerID = 1
)
```

Dans le cas ou on doit utiliser plusieurs fois la même table dans les SELECT imbriqués, on renomme les sous tables utilisées avec différents **ALIASES** :

```
SELECT * FROM personnes p1
WHERE p1.PerDateNaiss > (
    SELECT p2.PerDateNaiss FROM personnes p2
    WHERE p2.PerID = 1
)
```

Requêtes sélection multi-tables

```
SELECT * FROM livres, editeurs
WHERE  editeurs.EdiID = livres.LivEdiID
```

Fonctions de mysql (*Aggrégations*)

```
SELECT COUNT(personnes.PerID) AS "Nombre de personnes" FROM personnes
```

Note : Ici utilisation du mot clé **AS** pour changer le nom de la colonne.

```
SELECT MIN(personnes.PerDateNaiss) FROM personnes
```

```
SELECT personnes.PerNom FROM personnes
WHERE personnes.PerDateNaiss = (SELECT MAX(personnes.PerDateNaiss) FROM personnes)
```

Note : Ici, on a une requête imbriquée pour sélectionner l'enregistrement correspondant mais il n'y a pas besoin de renommer la table utilisée deux fois parce que on compare le max et pas un enregistrement spécifique !

```
SELECT SUM(livres.LivPrix) from livres
```

```
SELECT AVG(livres.LivPrix) from livres
```

Sélection avec regroupement

```
SELECT AVG(livres.LivPrix), livres.LivEdiID FROM livres
WHERE livres.LivNbPages > 150
GROUP BY livres.LivEdiID
HAVING AVG(livres.LivPrix) < 15
```

Note : Ici, on groupe par éditeurs. La clause **WHERE** est exécutée avant regroupement, la clause **HAVING** est exécutée séparément sur chaque groupe.

Jointure interne

```
SELECT * FROM livres
INNER JOIN emprunts ON livres.LivID = emprunts.EmpLivID
```

Jointures externe

```
SELECT * FROM livres LEFT JOIN emprunts ON livres.LivID = emprunts.EmpLivID
```

```
SELECT * FROM emprunts RIGHT JOIN livres ON livres.LivID = emprunts.EmpLivID
```

Note : les deux renvoie les mêmes données parce qu'on a inversé l'ordre des tables !

Requètes ensemblistes

```
SELECT personnes.PerPrenom FROM personnes
UNION
SELECT auteurs.AutPrenom FROM auteurs
```

```
SELECT personnes.PerPrenom FROM personnes
UNION ALL
SELECT auteurs.AutPrenom FROM auteurs
```

Note : **UNION ALL** affiche les doublons tandis qu'**UNION** n'afiche que les valeurs distinctes.

```
SELECT auteurs.AutNom  FROM auteurs
WHERE auteurs.AutNom IN (
    SELECT personnes.PerNom FROM personnes
)
```

Note : Le mot clé **IN** permet de faire l'*intersection* entre les deux tables.

```
SELECT auteurs.AutNom  FROM auteurs
WHERE auteurs.AutNom NOT IN (
    SELECT personnes.PerNom FROM personnes
)
```

Note : Le mot clé **NOT IN** permet de renvoyer la *Différence* entre les deux tables.

Miscellaneous

Fonction DATEDIFF(date1, date2)

```
SELECT DATEDIFF(auteurs.AutDateDeces, auteurs.AutDateNaiss)/365.25, auteurs.AutNom FROM auteurs
```

Notes :

- **DATEDIFF()** renvoit la différence en jours entre deux dates.
- Ici, on divise le total par 365.25 pour obtenir un résultat en années.

Fonction NOW()

```
SELECT now()
```

```
SELECT DATEDIFF(now(), personnes.PerDateNaiss)/365.25, personnes.PerNom FROM personnes
```

Note: Fonction **NOW()** renvoit la date et l'heure actuelle. On s'en sert ici pour calculer l'age des personnes.

fonction FLOOR(float)

```
SELECT FLOOR(DATEDIFF(auteurs.AutDateDeces, auteurs.AutDateNaiss)/365.25), auteurs.AutNom FROM auteurs
```

Note : **FLOOR** prend un float en en retourne l'entier inférieur. On l'utilise ici pour obtenir un nombre d'années sans rien après la virgule.

Opérateurs AND et OR

```
SELECT * FROM livres
WHERE livres.LivPrix < 15 AND livres.LivNbPages > 200
```

```
SELECT * FROM livres
WHERE livres.LivPrix < 15 OR livres.LivNbPages > 200
```

Opérateur IS NULL

```
SELECT * FROM livres
LEFT JOIN emprunts ON livres.LivID = emprunts.EmpLivID
WHERE emprunts.EmpID IS NULL
```

Note : L'utilisation de **IS NULL** est obligatoire et remplace le = pour vérifier si une valeur est nulle dans la db.

On peut cependant le remplacer par la fonction **ISNULL()** :

```
SELECT * FROM livres
LEFT JOIN emprunts ON livres.LivID = emprunts.EmpLivID
WHERE ISNULL(emprunts.EmpID)
```

Opérateur BETWEEN

```
SELECT * FROM livres
WHERE livres.LivNbPages BETWEEN 200 AND 250
```