29 – SQL BILAN ET RECAPITULATIF

I – Les trois relations de l'exercice

proposition (**id** INTEGER, **intitule** VARCHAR(10))

eleve (**id** INTEGER, **nom** VARCHAR(20), **prenom** VARCHAR(20), **naissance** DATE(20), **nsi** INTEGER, **moyenne** INTEGER)

voeu (**id** INTEGER, #**eleve_id** INTEGER, **description** VARCHAR(40), #**proposition_id** INTEGER, **reponse** INTEGER)

eleve_id est une clé étrangère menant à id de eleve
proposition_id est une clé étrangère menant à id de proposition

proposition	
id	**intitule**
1	Pas encore de réponse
2	Non
3	Liste d'attente
4	Oui mais
5	Oui

eleve					
id	**nom**	**prenom**	**naissance**	**nsi**	**moyenne**
1	Inbordeland	Alice	05/08/2002	12	14
2	Leponge	Bob	15/03/2003	6	13
3	Kent	Clark	30/01/2001	12	10
4	Wonderwoman	Diana	15/03/2001	18	18

voeu				
id	**#eleve_id**	**description**	**#proposition_id**	**reponse**
1	1	MP2I	3	NULL
2	1	Licence Info	5	NULL
3	2	Licence Océanographie	5	NULL
4	2	BUT Info	4	NULL
5	3	BUT Info	5	NULL
6	3	MP2I	1	NULL
7	4	MP2I	5	NULL
8	4	Licence Info	5	NULL

II – Rappels sur l'interrogation d'une BDD en SQL

SELECT a FROM b WHERE c ORDER BY d

1. **SELECT** : on choisit les attributs qu'on veut réellement récupérer dans les nuplets obtenus par FROM WHERE. On peut également faire des calculs (agrégation) sur ces nuplets avec MIN, MAX, AVG, SUM ou COUNT. L'utilisation de l'étoile * veut dire de récupérer l'intégralité des attributs disponibles sur les nuplets obtenus.
2. **FROM** : on indique la table dans laquelle on veut récupèrer des nuplets
3. **WHERE** : on fournit les conditions à respecter pour qu'un nuplet soit sélectionné. On pourra utiliser AND, OR, LIKE...
4. **ORDER BY** : on fournit l'attribut de tri avec un suffixe ASC ou DESC qui permettra d'ordonner les nuplets obtenus

```
1 SELECT nom, prenom
2 FROM eleve
3 WHERE naissance LIKE '%2001%'
4 ORDER BY nom DESC;
```

	nom	prenom
1	Wonderwoman	Diana
2	Kent	Clark

eleve					
id	**nom**	**prenom**	**naissance**	**nsi**	**moyenne**
1	Inbordeland	Alice	05/08/2002	12	14
2	Leponge	Bob	15/03/2003	6	13
3	Kent	Clark	30/01/2001	12	10
4	Wonderwoman	Diana	15/03/2001	18	18

01° Fournir la réponse obtenue avec cette requête :

```
SELECT nom, prenom, naissance
FROM eleve
WHERE prenom
LIKE '%i%' ORDER BY nom;
```

02° Fournir la réponse obtenue avec cette requête :

```
SELECT nom, prenom, naissance
FROM eleve
WHERE prenom
LIKE '%i%' OR naissance LIKE "%/03/%"
ORDER BY nom DESC;
```

03° Fournir la réponse obtenue avec cette requête :

```
SELECT DISTINCT eleve_id
FROM voeu
WHERE proposition_id = 5;
```

04° Quelle va être la différence la requête précédente ?

```
SELECT DISTINCT eleve_id AS "Numéro de l'élève"
FROM voeu
WHERE proposition_id = 5;
```

Requêtes imbriquées :

```
SELECT nom
FROM eleve
WHERE id IN
  (SELECT DISTINCT eleve_id AS "Numéro de l'élève"
  FROM voeu
  WHERE proposition_id = 5);
```

```
SELECT nom
FROM eleve
WHERE id IN
  (1, 2, 3, 4);
```

	nom
1	Inborderland
2	Leponge
3	Kent
4	Wonderwoman

III – Jointures avec JOIN ON

La clause JOIN ON permet de créer une table temporaire à l'aide de deux ou plusieurs autres tables.

Le principe est de faire la jonction à l'aide d'une comparaison entre l'attribut faisant office de clé étrangère et la clé primaire de l'autre table.

```
SELECT *
FROM table_a
JOIN table_b ON table_a.cle_etrangere = table_b.cle_primaire
WHERE condition(s)
ORDER BY expression
```

Avec JOIN ON : on indique la table qu'on veut fusionner avec la précédente et on précise **quelle clé primaire comparer à quelle clé étrangère.**

05° Remplir intuitivement la table suivante en utilisant les relations fournies en partie I.

id	#eleve_id	description	#proposition_id	reponse	nom de l'élève	proposition en fr
1	1	MP2I	3	NULL		
2	1	Licence Info	5	NULL		
3	2	Licence Océanographie	5	NULL		
4	2	BUT Info	4	NULL		
5	3	BUT Info	5	NULL		
6	3	MP2I	1	NULL		
7	4	MP2I	5	NULL		
8	4	Licence Info	5	NULL		

06° Fournir la requête comportant un JOIN ON permettant d'obtenir l'affichage suivant à partir des tables voeu et eleve uniquement.

	nom	description	proposition_id
1	Inborderland	MP2I	3
2	Inborderland	Licence Info	5
3	Leponge	Licence Océanographie	5
4	Leponge	BUT Info	4
5	Kent	BUT Info	5
6	Kent	MP2I	1
7	Wonderwoman	MP2I	5
8	Wonderwoman	Licence Info	5

07° Recopier et compléter la requête ci-dessous pour qu'elle fasse le travail demandé. On remarquera qu'elle possède deux jointures.

```
SELECT
  eleve.nom as "Nom de l'élève",
  voeu.description as "Formation",
  proposition.??? as "Réponse de la formation"
FROM voeu
JOIN eleve ON voeu.eleve_id = eleve.id
JOIN proposition ON voeu.??? = ???.???
```

	Nom de l'élève	Formation	Réponse de la formation
1	Inborderland	MP2I	Liste d'attente
2	Inborderland	Licence Info	Oui
3	Leponge	Licence Océanographie	Oui
4	Leponge	BUT Info	Oui mais
5	Kent	BUT Info	Oui
6	Kent	MP2I	Pas encore de réponse
7	Wonderwoman	MP2I	Oui
8	Wonderwoman	Licence Info	Oui

08° Quelle clause WHERE faire si on veut récupérer une description similaire à la précédente mais uniquement pour les voeux licence ?

```
SELECT
eleve.nom as "Nom de l'élève",
voeu.description as "Formation",
proposition.intitule as "Réponse de la formation"
FROM voeu
JOIN eleve ON voeu.eleve_id = eleve.id
JOIN proposition ON voeu.proposition_id = proposition.id
WHERE ???
```

	Nom de l'élève	Formation	Réponse de la formation
1	Inborderland	Licence Info	Oui
2	Leponge	Licence Océanographie	Oui
3	Wonderwoman	Licence Info	Oui

09° Quelqu'un veut obtenir la liste des élèves qui ont demandé une MP2I, n'ont pas été réfusés mais n'ont pas encore répondu. Voici sa requête :

```
SELECT eleve.nom
FROM voeu
JOIN eleve ON voeu.eleve_id = eleve.id
JOIN proposition ON voeu.proposition_id = proposition.id
WHERE voeu.description LIKE '%MP2I%'
  AND voeu.proposition_id != 2
  AND voeu.reponse = NULL;
```

Question : Qu'est-ce que ne fonctionne pas ? Comment formuler correctement la demande ?

On retrouve les choses habituelles pour relier plusieurs conditions entre elles et faire une condition plus complexe.

1. AND
2. OR
3. NOT
4. IS
5. IN
6. LIKE

IV – Commandes de modification

INSERT INTO

Cette commande permet de **rajouter des nuplets dans la table indiquée**.

```
INSERT INTO eleve
VALUES (NULL, 'Stark', 'Edward', '20/10/2002', NULL, NULL, NULL);
```

On notera que comme l'identifiant est défini automatiquement, sa valeur n'est pas à fournir : c'est la base de données qui va lui attribuer un numéro. C'est pour cela qu'on fournit NULL comme premier argument.

On peut également **ne donner que quelques éléments**, ou **en mettre plusieurs à la suite**.

```
INSERT INTO eleve (nom, prenom, naissance)
VALUES
('Sacquet', 'Frodon', '19/07/2003'),
('Grey', 'Jane', '15/03/2001');
```

UPDATE - SET

Cette commande permet de **modifier les valeurs des nuplets sélectionnés**.

Nous avons rentré deux nouveaux élèves mais ils n'ont pas de notes. On pourrait ainsi mettre 10 de base (à gauche) ou augmenter de 1 les moyennes de tous les élèves qui ont déjà au moins 10 en NSI (à droite).

```
1          UPDATE eleve                       UPDATE eleve
2          SET nsi = 10, moyenne = 10         SET nsi = nsi + 1
3          WHERE nsi IS NULL;                 WHERE nsi >= 10;
```

DELETE FROM

Cette commande permet de **supprimer les nuplets qui ont été sélectionnés**.

Attention : si vous ne placez pas de WHERE, vous allez supprimer l'intégralité des nuplets de la table qui va se retrouver totalement vide...

Pour détruire tous les nuplets d'une relation : **DELETE FROM voeu;**

Plus aucun voeu après ça. Laisser des droits de suppression à quelqu'un peut s'avérer dangereux...

Pour détruire les nuplets d'une relation répondant à certaines conditions :

```
1       DELETE
2       FROM voeu
3       WHERE proposition_id = 1
```

Avec cela, on supprime tous les voeux qui n'ont pas été acceptés par les formations.

10° Quelle requête faire pour rajouter l'élève parfait dans la base ?

11° Quelle requête pour parvenir à augmenter de 1 la moyenne de tous les élèves nés en 2001 ?

12° Comment parvenir à supprimer tous les élèves dont le nom contient un A ?

C'est assez complexe. Pour cela, il faudra :

- Faire une première requête SELECT permettant de récupérer tous les id d'élèves dont le nom contient un A.
- Détruire les nuplets de voeu dont l'identifiant élève est dans (IN) la réponse précédente

V – Fonctions d'agrégation

Lorsqu'on récupére des entrées, on peut les utiliser pour effectuer certaines recherches ou calculs :

- **MAX** pour trouver la valeur la plus grande sur cet attribut dans la sélection obtenue
- **MIN** pour trouver la valeur la plus petite sur cet attribut dans la sélection obtenue
- **AVG** pour trouver la valeur moyenne sur cet attribut dans la sélection obtenue
- **COUNT** pour trouver le nombre de nuplets obtenus dans la sélection

13° Que fait cette requête ?

```
1      SELECT COUNT(id)
2      FROM voeu
3      WHERE voeu.description LIKE '%IUT%';
```

14° Que fait cette requête ?

```
1  SELECT COUNT(DISTINCT eleve.nom)
2  FROM eleve
3  JOIN voeu ON eleve.id = voeu.eleve_id
4  WHERE voeu.description LIKE '%IUT%';
```

15° Fournir la requête permettant d'obtenir la moyenne en NSI des élèves de la base.

16° Fournir la requête permettant d'obtenir la moyenne en NSI des élèves de la base qui ont été acceptés dans une formation avec une proposition_id valant 5. Encore une fois, nous avons besoin à la fois de la table eleve et de la table voeu.