**RÉVISION** :

Les symboles utilisés dans les requêtes SQL :

| \* | sélectionne toutes les colonnes; tout |
| --- | --- |
| = / LIKE | égale |
| != | différent |
| <,<=,>,<= | inférieur à, inférieure ou égal à ; supérieur, supérieur ou égal à |
| [ ] | crochet , ex : [1-4]\* ( un chiffre qui commence par un chiffre entre 1 et 4. |
| % | un caractère qui remplace tous les autres caractères (ex : ‘%a’ - termine par a, ‘a%’-commence par a , ‘%a%’ - contient un a) |
| \_ | peut être remplacé un seul caractère |
| ^ |  |
| + ; -;\*;/ | opérateurs : plus ; moins, multiplication, division |

Les opérateurs + leur syntaxe (un exemple par opérateur )

**OPÉRATIONS basiques :**

SELECT nom\_colonne

FROM nom\_table

**WHERE** prédicat (condition)

SELECT DISTINCT - Pour éviter des redondances dans les résultats

pour trier et mettre en ordre les résultats :

SELECT nom\_colonne

FROM nom\_table

**ORDER BY** Artibut (DESC-ordre décroissant ; ASC-ordre croissant)

Pour appliquer la fonction d'agrégation sur plusieurs tuples et trier par champ /regroupement.

SELECT nom\_colonne

FROM nom\_table

**GROUP BY** Artibut

**HAVING** Condition

WHERE : filtre sur les données des tables

HAVING : filtre sur les résultats des calculs de la requête.

**Jointure** entre relations :

1e

**SELECT** liste des attributs

**FROM** table1, table 2

**WHERE** table1.attribut1 = table2.attribut2

2e

**SELECT** attributs

**FROM** Table1 INNER JOIN Table2

**ON** Table1.atttribut1 = Table2.attribut2

**1. Qui sont les professeurs qui donnent des cours mardi et quels sont leurs cours?**

**SELECT nom\_prof, nom\_cours**

**FROM prof INNER JOIN cours**

**ON prof.num\_cours = cours.nom\_cours**

**WHERE jour = ‘mardi’ ;**

**2. Combien de participants y a-t-il par cours, dans quelles salles et quels gymnases?**

**SELECT COUNT(num\_participant),salle,num\_gymnase**

**FROM inscription, cours**

**WHERE inscription.num\_cours = cours.num\_cours**

**GROUPE BY nom\_cours ;**

**3. Quels sont les cours les plus populaires et quels sont leurs enseignants ?**

**SELECT nom\_cours, nom\_prof**

**FROM cours, prof**

**WHERE cours.num\_cours = prof.num\_cours**

**ORDRE BY ASC(place\_libre)**

**1. Quels sont les cours qui n’ont pas lieu à la piscine des pompiers**

**SELECT num\_cours, nom\_cours**

**FROM cours INNER JOIN gymnase**

**ON cours.num\_gymnase = gymnase.num\_gymnase**

**WHERE nom\_gymnase !=’piscine des pompiers’ ;**

**Union** : (concaténation des deux tables, élimination des doublons)

1e

**SELECT** attributs

**FROM** table1

**UNION**

**SELECT** attributs

**FROM** table2

2e

concaténation des relations avec les nouveaux attribut :

**SELECT** attribut AS( à ajouter les guillemet) ??? attribut1, attribut1 AS attribut2

**FROM** table1

**UNION**

**SELECT** attribut AS attribut1,attribut1 AS attribut2

**FROM** table2

3e : (ajout des conditions )

**SELECT** attribut

**FROM** table1

**WHERE** Condition

**UNION**

**SELECT** attribut

**FROM** table2

**WHERE** Condition

4e : (pour garder les doublons en cas de besoin)

**SELECT** attributs

**FROM** table1

**UNION ALL**

**SELECT** attributs

**FROM** table2

5e : ( création d’une nouvelle table)

**INSERT INTO table3**

**SELECT \***

**FROM (SELECT** attributs

**FROM** table1

**UNION**

**SELECT** attributs

**FROM** table2**)**

**Intersection : (**parties communes de deux tables)

1e

**SELECT** attributs

**FROM** table1

**WHERE** attribut1 **IN**

**(SELECT** attribut1 **FROM** table 2**);**

2e **:**

**SELECT** attributs

**FROM** table1

**INTERSECT**

**SELECT** attributs

**FROM** table2

**Différence (**les éléments dans table1 mais pas dans table2, visé versa**)**

1e

**SELECT** attributs

**FROM** table 1

**WHERE** attribut 1 **NOT IN**

**(SELECT** attribut1 **FROM** table 2**);**

2e

**SELECT** attribut 1, attribut 2…. **FROM** table 1

**EXCEPT**

**SELECT** attribut 1, attribut 2…**. FROM** table 2**;**

**LEFT JOIN**

**SELECT \***

**FROM A LEFT JOIN B ON A.key = B.key**

**LEFT JOIN (sans l’intersection de B)**

**SELECT \***

**FROM A LEFT JOIN B**

**ON A.key = B.key WHERE B.key IS NULL**

**Fonctions d'agrégation :**

**SELECT Fonction d'agrégation (**nom\_attribut**)**

**SUM**(nom\_attribut) : calcule la somme des valeurs des

attributs (type numérique)

**MIN**(nom\_attribut) : recherche la plus petite valeur de l’attribut

**MAX**(nom\_attribut) : recherche la plus grande valeur de

l’attribut

**COUNT**(nom\_attribut) : compte le nombre d’occurrences de l’attribut

**AVG**(nom\_attribut) : calculer la moyenne des valeurs des attributs (type numérique)

Remarque :Possibilité deavec le mot clé **AS (alias pour renommer temporairement une colonne ou une table dans une requête)**

**Opérateurs :**

**LIKE(=)**

**WHERE** xxx **LIKE** xxx **;**

**NULL (**champ vide**)**

**WHERE** xxx **IS (NOT) NULL**

**IN (**colonne égale à valeur1 ou valeur2 ou…**)**

**WHERE** xxx **IN (**valeur1, valeur2,,...**)**

**WHERE** xxx **NOT IN** xxx

**BETWEEN** xxx **AND** xxx **(**sélectionner un intervalle de données **)**

**WHERE** xxx **BETWEEN** valeur1 **AND** valeur2

**OR (**Ou**)**

**WHERE** Condition1 **OR** Condition2

**WHERE** Condition1

**GROUP BY (**Regroupement de Tuples pour une valeur d'attributs**)**

**Note :** Cette commande doit toujours s’utiliser **après la commande WHERE** et **avant la commande HAVING.**

**HAVING** condition2

**SELECT / FROM xxx AS xx :** (renommer temporairement une colonne ou une table dans une requête)

**Sous-requêtes**

**EXISTS** vérifie si la sous-requête retourne un résultat ou non

**SELECT \***

**FROM** table

**WHERE EXISTS (**SELECT colonne

FROM table2

WHERE condition**) ;**

**IN** vérifie la concordance d’une à plusieurs données

**SELECT \***

**FROM** table

**WHERE** colonne **(**SELECT colonne

FROM table2

WHERE condition**) ;**

**ANY (=IN) ; SOME :**comparer une valeur avec le résultat d’une sous-

requête. Il est ainsi possible de vérifier si une valeur est “égale”

**SELECT** colonne

**FROM** table

**WHERE** colonne **> ANY (SELECT** colonn**e**

**FROM** table2**);**

**1. Quels sont les cours dont la durée est égale à au moins un cours enseigné par « Nicolas Dumas » ?**

**SELECT num\_cours,nom\_cours**

**FROM cours**

**WHERE durée = ANY(SELECT durée**

**FROM cours**

**INNER JOIN prof ON cours.num\_prof = prof.num\_prof**

**WHERE prénom = ‘Nicolas’ AND nom = ‘Dumas’);**

**ALL**

comparer une valeur dans l’ensemble de valeurs d’une sous-requête

**SELECT** colonne**1**

**FROM** table**1**

**WHERE** colonne1 **> ALL (SELECT** colonne1

**FROM** table2

**WHERE** condition **);**

**Créer une BDD**

**CREATE DATABASE** (**IF NOT EXISTS**) nom\_base ;

**Supprimer une BDD**

**DROP DATABASE** (**IF NOT EXISTS**)nom\_base ;

**Supprimer des lignes**

DELETE FROM nom\_table

WHERE predicat

**Renommer une table**

**RENAME** ancien\_nom **TO** nouveau\_nom ;

**Ouvrir une BDD**

**START DATABASE** nom\_base ;

**Fermer une BD**

**STOP DATABASE** ;

**Créer une table**

**CREATE TABLE** nom\_table

(nom\_col1 TYPE1,

(nom\_col2 TYPE2,

…)

TYPE : string, date, entiers, et

**Déclaration des clés**

CREATE TABLE `nom\_table` (

id INT NOT NULL PRIMARY KEY AUTO\_INCREMENT) ;

**Ajout / suppression d’une colonne**

**ALTER TABLE** nom\_table **ADD** nom\_colonne type\_donnees;

**ALTER TABLE** nom\_table **DROP** (**COLUMN**) nom\_colonne;

**Suppression de lignes**

**DELETE FROM** nom\_table

**WHERE** prédicat ;

**Modification/Renommer d’une colonne**

**ALTER TABLE** nom\_table **MODIFY** nom\_colonne type\_donnees ;

**ALTER TABLE** nom\_table **CHANGE** ancien\_nom nouveau\_nom type\_donnees ;

**Ajout de tuples (lignes)**

**INSERT INTO** nom\_table (nom\_col1, ...) **VALUES** (val1, …);

ou

**INSERT INTO** nom\_table **VALUES** (value1, value2, ... ) - plus simple

**INSERTION DES LIGNES D’UNE AUTRE TABLE**

**INSERT INTO** nom\_table

**SELECT** colonne

**FROM** nom\_table

**WHERE** condition ;

**Update d’une base : lignes**

**UPDATE** nom\_table

**SET** nom\_col1= ‘’,

nom\_col2= ‘’, …

**WHERE** predicate;

ex :

UPDATE client

SET pays = 'FRANCE‘;

8. Je suis Lison Martin, qui a eu de meilleures notes que moi et en quel sport?

SELECT participant.nom,participant.prénom,cours.nom\_cours,inscription.score

FROM inscription i

LEFT JOIN participant p on i.num\_participant = p.num\_participant

LEFT JOIN cours c on i.num\_cours = c.num\_cours

LEFT JOIN

(SELECT distinct num\_cours, score from inscription

WHERE num\_participant =

(SELECT num\_participant from participant

WHERE nom= “Martin” AND prénom= “Lison”)) lms

ON lms.num\_cours=i.num\_cours

WHERE lms.num\_cours=i.num\_cours

AND i.score>lms.score

9. Quelle requête peut donner le résultat suivant : (2 points)

SELECT \*,g.nom\_gymnase,g.metro

(SELECT c.Num\_cours, c.nom\_cours,

AVG(2020-cast(substr(p.date\_naissance,1,4) as int)) as age\_moyenne,

AVG(score) as note\_moyenne,

(COUNT(p.sexe=‘F’)/count(\*)\*100) as pourcentage\_des\_femmes,

COUNT(i.num\_participant) as nombre\_participant,num\_gymnase

FROM cours c, inscription i

LEFT JOIN participant p ON i.num\_participant = p.num\_participant

LEFT JOIN cours c ON i.num\_cours = c.num\_cours

GROUP BY i.num\_cours,c.nom\_cours,num\_gymnase

) db

LEFT JOIN gymnase g ON g.num\_gymnase = db.num\_gymnase

ORDER BY db.num\_cours;

#la fonction(substr()), cela veut dire segementer une partie de string.Répondre aux questions suivantes (toutes les questions sont indépendantes) en

exprimant la requête en langage SQL :

1. Quels sont les exemples d’emploi du lexème « monter » ?

SELECT Emploi.exemple

FROM Emploi

INNER JOIN Lexemes

ON Lexemes.numero\_lexeme=Emploi.numero\_lexeme

WHERE Lexemes.lexeme LIKE “monter”;

2. Quels sont les lexèmes qui n’appartiennent à aucune classe de synonymes ?

SELECT Lexemes.lexeme

FROM Lexemes

INNER JOIN Classe\_Synonymes

ON Lexemes.numero\_lexeme=Classe\_Synonymes.numero\_lexeme

WHERE Classe\_Synonymes.numero\_lexeme=

(SELECT Classe\_Synonymes.numero\_lexeme FROM Classe\_Synonymes WHERE

Classe\_Synonymes.nom\_classe\_synonyme IS NULL);

3. Quel est le nom de la classe qui a pour classe antonyme la classe de nom

« MONTER » ?

SELECT nom\_classe

FROM Antonymie

WHERE nom\_classe\_antonyme LIKE “MONTER”;

4. Quels sont les numero\_lexeme qui appartiennent à la classe qui a pour classe

antonyme la classe de nom « HAIR » ?

SELECT Lexemes.numero\_lexeme

FROM Lexemes

INNER JOIN Antonymie

ON Antonymie.numero\_parangon\_de\_nom\_classe=Lexemes.numero\_lexeme

WHERE Antonymie.nom\_ classe\_antonyme LIKE “HAIR”;

5. Quels sont les lexèmes qui appartiennent à la classe qui a pour classe

antonyme la classe de nom « HAIR » ? (2 points)

SELECT Lexemes.lexeme

FROM Lexemes

INNER JOIN Antonymie

ON Antonymie.numero\_parangon\_de\_nom\_classe=Lexemes.numero\_lexeme

WHERE Antonymie.nom\_ classe\_antonyme IN

(SELECT Antonymie.nom\_ classe\_antonyme FROM Antonymie WHERE

Antonymie.nom\_ classe\_antonyme LIKE “HAIR” ;6. Quel est le registre de langue du parangon de la classe des synonymes de nom

« AIMER »

SELECT Lexemes.registre\_langue

FROM Lexemes

INNER JOIN Classe\_Synonymes

ON Lexemes.numero\_lexeme=Classe\_Synonymes.numero\_lexeme

WHERE Classe\_Synonymes.nom\_classe\_synonyme LIKE “AIMER”;

7. Quelle est la définition du parangon de la classe des synonymes de nom

« AIMER » ?

SELECT Lexemes.intitule\_Definition

FROM Lexemes

INNER JOIN Classe\_Synonymes

ON Lexemes.numero\_lexeme=Classe\_Synonymes.numero\_lexeme

WHERE Classe\_Synonymes.nom\_classe\_synonyme LIKE “AIMER”;

8. Quel est le nom de la classe de synonyme auquel appartient le lexème « siège »

SELECT Classe\_Synonymes.nom\_classe\_synonyme

FROM Classe\_Synonymes

INNER JOIN Lexems

ON Lexemes.numero\_lexeme=Classe\_Synonymes.numero\_lexeme

WHERE Lexemes.lexeme LIKE “siège”;

9. Quel est le lexème qui est le parangon de la classe synonyme qui a pour nom

« HAIR »

SELECT Lexemes.lexeme

FROM Lexemes

INNER JOIN Classe\_Synonymes

ON Lexemes.numero\_lexeme=Classe\_Synonymes.numero\_lexeme

WHERE Classe\_Synonymes.nom\_classe\_synonyme LIKE “HAIR”;