13 SQL

1 Différents types

1.1 Entiers

TINYINT	- 128 à 127
SMALLINT	-3278 à 3277
MEDIUMINT	-8388608 à 8388607
INT	-2147483648 à 2147483647
BIGINT	-9223372036854775808 à + 9223372036854775807

1.2 Flottants

FLOAT	Nombre flottant, précis sur 23 chiffres
DOUBLE	Nombre flottant de 24 à 53 chiffres
DECIMAL	Type « double », mais stocké en tant que STRING

1.3 Chaines de caractères

CHAR(x)	Le nombre entre () définira l'espace qu'il prendra
VARCHAR(x)	Prendra 1 octet par lettre +1, dans la limite de (x)
TINYTEXT	Caractères limités par le nombre d'octet
TEXT	Caractères limités par le nombre d'octet
MEDIUMTEXT	Caractères limités par le nombre d'octet
LONGTEXT	Caractères limités par le nombre d'octet

1.4 Dates

DATE	YYYY-MM-DD
DATETIME	YYYY-MM-DD Hh:mm:ss
TIME	Hh:mm:ss
YEAR	YYYY
TIMESTAMP	YYYYMMDDHhmmss

2 CONVENTIONS SQL

2.1 Nom de tables et colonnes

- Pas d'espace (ils sont remplacés par des underscores)
- Pas de caractères spéciaux
- Pas de mots réservés (Ex : DATE, TEXT, TYPE, TABLE, etc.)
- Noms représentatifs, sans abréviations
- Privilégier le singulier

2.2 Syntaxe des commandes

- Mots-clés en majuscules
- Noms des bases, des tables et des colonnes en minuscules
- Options facultatives des commandes entre crochets

3 CONNEXION À MARIADB / POSTGRESQL

- **Se connecter à MariaDB en local :** mariadb -u root -p
- **Se connecter au serveur MariaDB distant :** mariadb -h serveur_distant -u user -p pwd
- **Se connecter à PostGreSQL :** psql -U postgres

4 MANIPULATIONS DE BASE

- **Effacer l'écran du terminal** (sur Windows) **:** \! cls
- **Écrire un commentaire :** --commentaire **OU :** # commentaire

4.1 Commandes exclusives à MariaDB

- **Connaitre la base de données courante :** SELECT DATABASE();
- **Afficher les bases de données existantes :** SHOW DATABASES;
- **Afficher les bases de données existantes contenant une certaine chaine de caractères :** SHOW DATABASES like '%chaine%';
- **Sélectionner une des bases de données** ou **changer la base de données courante :** USE bdd;
- **Afficher les tables d'une base de données :** SHOW TABLES;
- **Afficher les caractéristiques d'une table (structure) :** DESCRIBE table; **OU :** DESC table;

- **Afficher le nom de l'utilisateur connecté :** SELECT USER();
- **Afficher la date courante** (du serveur) **:** SELECT CURDATE();
- **Quitter / stopper la connexion :** EXIT; **OU :** QUIT;
- **Vérifier si le cache est activité ou désactivé :** SHOW VARIABLES LIKE '%query_cache%';
- **Récupérer le nombre de lignes rapporté par la dernière requête :** FOUND_ROWS()
- **Convertir une variable en un certain type de données :** CAST(variable AS TYPE)

4.2 Commandes exclusives à PostGreSQL

- **Afficher la liste de toutes les commandes :** \?
- **Afficher la liste des bases de données :** \l
- **Changer la base de données courante :** \connect bdd, \c bdd
- **Afficher des informations sur la connexion en cours :** \conninfo
- **Afficher la version de PostGreSQL :** SELECT VERSION();
- **Afficher la date courante** (du serveur) **:** SELECT CURRENT_DATE;
- **Afficher l'heure courante** (du serveur) **:** SELECT CURRENT_TIME;
- **Afficher les tablespaces existants :** \db
- **Afficher plus d'informations sur les tablespaces existants :** \db+
- **Afficher les tables d'une base de données :** \d;
- **Afficher les caractéristiques d'une table (structure) :** \d table;
- **Retarder l'exécution du processus serveur de 5 secondes :** PG_SLEEP(5);
- **Quitter :** \q

5 UTILISATION DE FICHIERS

5.1 MariaDB

- **Sauvegarder une copie des commandes que l'on va écrire dans un fichier :** TEE chemin/fichier.txt;
- **Arrêter l'enregistrement des commandes dans le fichier :** NOTEE;
- **Exécuter un script SQL sur la console :** SOURCE chemin/fichier.sql;

5.2 PostGreSQL

• **Sauvegarder une copie des sorties que l'on va afficher dans un fichier :** \o chemin/fichier.txt;

• **Arrêter l'enregistrement des sorties dans le fichier :** \o;

• **Exécuter un script SQL sur la console (Linux) :** psql -U postgres -f chemin/fichier.sql;

• **Exécuter un script SQL sur la console (Windows) :** psql -U postgres \i chemin/fichier.sql;

6 SELECT … FROM … WHERE …

6.1 SELECT

• **Sélectionner tous les champs :** SELECT *

• **Sélectionner un champ en supprimant ses doublons :** SELECT DISTINCT champ

• **Renommer un champ :** SELECT champ AS "champ_renommé"

• **Préfixer un champ** (pour prévenir le risque d'ambigüité) **:** SELECT table.champ

Remarque : dès que l'on utilise une opération ou une fonction, il faut utiliser un alias de colonne.

6.2 FROM

• **Renommer une table :** table AS table_renommée

• **Combiner les données de plusieurs tables :**

SELECT table1.colonne1, table2.colonne2

FROM table1 JOIN table2 ON table1.colonne_clé1 = table2.colonne_clé2

— INNER JOIN : renvoie l'informatique quand elle est vraie dans les deux tables

— LEFT/RIGHT JOIN : renvoie toutes les colonnes de la table de gauche/droite et, dans les colonnes de la table droite/gauche, affiche NULL s'il n'y a pas de données

WHERE table1.colonne1 > valeur;

6.3 WHERE

• **Définir un intervalle :** champ BETWEEN borne1 AND borne2

• **Comparer un champ à une liste de valeurs :** champ (NOT) IN(valeur1, valeur2, valeur3)

• **Comparer partiellement une chaine :** champ (NOT) LIKE '%chaine_' ; champ (NOT) LIKE 'ch___aine'

Remarques :

— le symbole % remplace zéro, un ou plusieurs caractères

— le symbole _ remplace un caractère

— utiliser le caractère d'échappement « \ » pour utiliser les caractères % ou _ dans la requête

• **Combiner plusieurs conditions :**

— **et retourner vrai si les deux conditions sont vraies :** (condition1) AND (condition2)

— **et retourner vrai si l'une au moins des conditions est vraie :** (condition1) OR (condition2)

— **et retourner vrai si la condition qui suit l'opérateur est fausse :** (condition1) NOT (condition2)

Ordre de priorité : parenthèses > NOT > AND > OR.

• **Vérifier si une valeur est nulle ou pas :** valeur IS (NOT) NULL

• **Évaluer une sous-requête :**

— **comparer un champ aux résultats d'une sous-requête :** champ IN (sous-requête)

— **comparer un champ à chaque résultat d'une sous-requête :** champ ANY (sous-requête)

— **comparer un champ à tous les résultats d'une sous-requête :** champ ALL (sous-requête)

6.4 GROUP BY

• **Organiser des données identiques en groupes** (« en fonction de champ1 et de champ2 », « pour chaque champ1 et chaque champ2 ») **:** GROUP BY champ1, champ2

• Remarque : Dès que l'on souhaite afficher plusieurs champs dont l'un est composé d'une fonction de calcul, il faut effectuer un regroupement.

Toute colonne ou expression de la liste SELECT autre qu'une fonction de groupe doit être incluse dans la clause GROUP BY.

6.5 HAVING

• **Filtrer des agrégations :** HAVING condition

Ex : SELECT MAX(champ)

FROM table

HAVING MAX(champ) > valeur/(sous-requête)

6.6 ORDER BY

• **Afficher le résultat dans un ordre croissant, alphabétique, chronologique :** ORDER BY champ1 ASC, champ2 ASC

• **Afficher le résultat dans l'ordre inverse :** ORDER BY champ DESC

6.7 LIMIT

• **Limiter le nombre de lignes à afficher :** LIMIT nombre_de_ligne_à_afficher OFFSET numéro_première_ligne_à_afficher

7 INSERT, UPDATE, DELETE

• Insérer de nouvelles lignes dans une table :

INSERT INTO table(champ1, champ2, …, champN)

VALUES(valeur1, valeur2, …, valeurN), (valeurA, valeurB, …, valeurX);

Remarques :

— On peut utiliser les valeurs spéciales comme « SYSDATE » qui renvoie la date et l'heure courantes.

— On peut utiliser des variables de substitution en écrivant : « &valeur ».

• Copier des lignes d'une autre table :

INSERT INTO table1(champ1, champ2)

SELECT champ1, champ2

FROM table2

WHERE condition;

• Modifier des données d'une table :

UPDATE table

SET champ1 = valeur1, champ2 = valeur2, …, champN = valeurN

WHERE condition;

• Modifier des lignes d'une table en fonction d'une autre table :

UPDATE table

SET champX = (sous-requête1)

WHERE champY = (sous-requête2);

• Supprimer des lignes d'une table :

DELETE FROM table

WHERE condition;

• Supprimer des lignes en faisant référence à une autre table :

DELETE FROM table

WHERE champ = (sous-requête);

Remarque : Une sous-requête ressemble à : SELECT champ FROM table WHERE condition

8 CREATE, ALTER, DROP

8.1 Bases de données

- **Créer une base de données :** CREATE DATABASE bdd;
- **Supprimer une base de données :** DROP DATABASE bdd;

8.2 Tables

- **Créer une table :** CREATE TABLE table(colonne1 type_données, colonne2 type_données);
- **Renommer une table :** ALTER TABLE table RENAME TO nouveau_nom;
- **Supprimer une(des) table(s) :** DROP TABLE table1, table2;

8.3 Colonnes

- **Ajouter une colonne :** ALTER TABLE table ADD colonne type_données;
- **Modifier une colonne :** ALTER TABLE table MODIFY colonne type_données;
- **Renommer une colonne :** ALTER TABLE table RENAME COLUMN colonne TO nouveau_nom;
- **Supprimer une colonne :** ALTER TABLE table DROP colonne;

8.4 Index

- **Créer un index sur une(des) colonne(s) :** CREATE INDEX idx_colonne ON table(colonne1, colonne2);
- **Supprimer un index :** DROP INDEX idx_colonneS ON table(colonne);

Remarque : les deux index par défaut sont « UNIQUE » et « PRIMARY KEY ».

8.5 Vues

- **Créer une vue :** CREATE VIEW v_vue AS

  SELECT colonne1, colonne2

  FROM table

  …

- **Modifier une vue :** CREATE OR REPLACE VIEW v_vue AS (pareil que pour la création)
- **Extraire les données d'une vue :** SELECT * FROM v_vue;
- **Supprimer une vue :** DROP VIEW v_vue;

Remarque : On peut utiliser « ALGORITHM = UNDEFINED/MERGE/TEMPTABLE » entre « CREATE (OR REPLACE) » et « VIEW vue » :

— MERGE : la vue doit être mise à jour lorsque la table sous-jacente est mise à jour (la vue est donc modifiable)

— TEMPTABLE : la vue utilise une table temporaire pour stocker les résultats intermédiaires, plutôt que d'effectuer des calculs en mémoire (la vue n'est pas modifiable)

— UNDEFINED : option par défaut si on ne précise pas d'algorithme. Ce sera le plus souvent MERGE mais certains éléments rendront une table incompatible avec MERGE (DISTINCT, LIMIT, fonctions d'agrégation, HAVING, UNION, une sous-requête dans la clause SELECT)

8.6 Séquences

• **Séquence :** permet de générer automatiquement des numéros uniques.

• **Créer une séquence :**

CREATE SEQUENCE séquence

START valeur_départ

INCREMENT valeur_incrémentale

MINVALUE valeur_minimum

MAXVALUE valeur_maximum

OWNED BY table.colonne (associe une séquence à la colonne d'une table)

• **Insérer de nouvelles lignes dans une table avec une valeur incrémentée par une séquence** (MariaDB) **:**

INSERT INTO table(champ1, champ2, …, champN)

VALUES(valeur1, NEXT VALUE FOR séquence, …, valeurN), (valeurA, NEXT VALUE FOR séquence, …, valeurX);

• **Insérer de nouvelles lignes dans une table avec une valeur incrémentée par une séquence** (PostGreSQL) **:**

INSERT INTO table(champ1, champ2, …, champN)

VALUES(valeur1, nextval(séquence), …, valeurN), (valeurA, nextval(séquence), …, valeurX);

• **Afficher la valeur actuelle d'une séquence** (MariaDB) **:** SELECT PREVIOUS VALUE FOR séquence;

• **Afficher la valeur actuelle d'une séquence** (PostGreSQL) **:** SELECT CURRVAL(séquence);

• **Supprimer une séquence :** DROP SEQUENCE séquence;

9 CONTRAINTES

• **Imposer à un champ qu'à tout enregistrement de la table, il soit rempli :** NOT NULL

Syntaxe : CREATE TABLE table (colonne type_données NOT NULL);

ALTER TABLE table ALTER COLUMN colonne SET NOT NULL;

• **S'assurer que toutes les valeurs d'un champ sont uniques et distinctes :** UNIQUE

Syntaxe :

CREATE TABLE table (… CONSTRAINT UQ_colonne NOT NULL (colonne), …);

ALTER TABLE table ADD COLUMN colonne type_données UNIQUE;

ALTER TABLE table ADD CONSTRAINT UQ_colonne UNIQUE (colonne);

ALTER TABLE table DROP INDEX UQ_colonne;

• **Cumuler les contraintes de NOT NULL et UNIQUE et créer le moyen de référence à un et un seul enregistrement dans une table :** PRIMARY KEY

Syntaxe :

CREATE TABLE table (… colonne type_données PRIMARY KEY, …);

CREATE TABLE table (… CONSTRAINT PK_colonne PRIMARY KEY (colonne1, colonne2));

ALTER TABLE table ADD CONSTRAINT PK_colonne PRIMARY KEY (colonne1, colonne2);

ALTER TABLE table DROP CONSTRAINT PK_colonne;

• **Mettre en place des liaisons entre les tables** (la clé étrangère est un champ faisant référence à un champ d'une autre table dont il est clé primaire) **:** FOREIGN KEY

Syntaxe :

CREATE TABLE table1 (… FOREIGN KEY (colonne1) REFERENCES table2(colonne2));

CREATE TABLE table (… CONSTRAINT FK_table1table2 FOREIGN KEY (colonne1) REFERENCES table2(colonne2));

ALTER TABLE table1 ADD FOREIGN KEY (colonne1) REFERENCES table2(colonne2);

ALTER TABLE table DROP FOREIGN KEY FK_colonne;

• **Attribuer une valeur par défaut à un champ lorsqu'elle n'est pas spécifiée pour ne pas avoir de champs vides :** DEFAULT

Syntaxe :

CREATE TABLE table (… colonne type_données DEFAULT valeur_par_défaut,…);

ALTER TABLE table ALTER COLUMN colonne SET DEFAULT valeur_par_défaut;

ALTER TABLE table ALTER COLUMN colonne DROP DEFAULT;

• **S'assurer que les données respectent un format bien précis, appartiennent à une liste de valeurs ou respectent une règle de gestion :** CHECK

Syntaxe :

CREATE TABLE table (… colonne type_données CHECK (condition));

CREATE TABLE table (… CONSTRAINT CHK_contrainte CHECK (condition));

ALTER TABLE table ADD CONSTRAINT CHK_contrainte CHECK (condition);

ALTER TABLE table DROP CONSTRAINT CHK_contrainte;

Exemples de condition : age>= 18 AND city='Sandnes' ; dateLivraison < datePeremption ; idPays ~ '[A-Z][A-Z][A-Z]' ; value>0

10 COMMIT, ROLLBACK

• **Valider les modifications réalisées depuis le début de la transaction :** COMMIT;

• **Annuler les modifications réalisées sur la base de données jusqu'à obtenir l'état qu'elle avait avant le début de la transaction :** ROLLBACK;

• **Créer un point de sauvegarde :** SAVEPOINT point_sauvegarde;

• **Annuler les modifications réalisées sur la base de données jusqu'à obtenir l'état qu'elle avait lors de la création du point de sauvegarde :** ROLLBACK TO SAVEPOINT point_sauvegarde;

11 FONCTIONS SUR LES CHIFFRES

• **Compter tout, sauf les valeurs NULL :** COUNT(*)

• **Compter le nombre de lignes non vides d'une colonne :** COUNT(colonne)

• **Compter le nombre de lignes différentes d'une colonne :** COUNT(DISTINCT colonne)

• **Renvoyer la plus petite valeur de toutes les valeurs sélectionnées d'une colonne :** MIN(colonne)

• **Renvoyer la plus grande valeur de toutes les valeurs sélectionnées d'une colonne :** MAX(colonne)

• **Renvoyer la moyenne des valeurs d'une colonne :** AVG(colonne)

• **Renvoyer la somme de toutes les valeurs d'une colonne :** SUM(colonne)

• **Renvoyer la somme des valeurs d'une colonne sans compter les valeurs dupliquées :** SUM(SELECT colonne)

• **Renvoyer la variance des valeurs d'une colonne :** VARIANCE(colonne)

• **Renvoyer l'écart-type des valeurs d'une colonne :** STDDEV(colonne)

• **Arrondir à l'entier supérieur :** CEIL(nombre)

• **Arrondir à l'entier inférieur :** FLOOR(nombre)

• **Arrondir au plus proche :** ROUND(nombre)

• **Arrondir au plus proche à la décimale près :** ROUND(nombre, décimale)

• **Tronquer à un certain nombre de décimale :** TRUNCATE(nombre, décimale)

• **Multiplier un nombre par lui-même un certain nombre de fois :** POW(nombre, fois)

• **Diviser un nombre par lui-même :** SQRT(nombre)

• **Indiquer en booléen si le nombre est positif (1) ou négatif (0) :** SIGN(nombre)

• **Renvoyer la valeur absolue (sans le signe négatif) :** ABS(nombre)

• **Renvoyer le modulo d'un nombre divisé par un autre nombre :** MOD(nombre, autre_nombre)

12 FONCTIONS SUR LES CHAINES

• **Transformer tous les caractères d'une chaine de caractères en minuscules :** LOWER(chaine)

• **Transformer tous les caractères d'une chaine de caractères en majuscules :** UPPER(chaine)

• **Indiquer le nombre de bits d'une chaine :** BIT_LENGTH(chaine)

• **Indiquer le nombre de caractères :** CHAR_LENGTH(chaine)

• **Retourner le nombre d'octets :** LENGTH(chaine)

• **Répéter une chaine de caractères un certain nombre de fois :** REPEAT(chaine, fois)

• **Remplir une chaine de caractères à gauche/droite jusqu'à une longueur spécifiée :** LPAD/RPAD(chaine, longueur, caractère_pour_remplir) (tronque la chaine si la longueur est inférieure au nombre de caractères de la chaine)

• **Supprimer les espaces au début et à la fin d'une chaine :** TRIM(chaine)

• **Supprimer les espaces uniquement au début d'une chaine :** TRIM(LEADING FROM chaine)

• **Supprimer les espaces uniquement à la fin d'une chaine :** TRIM(TRAILING FROM chaine)

• **Supprimer une sous-chaine au début et à la fin d'une chaine :** TRIM(sous-chaine FROM chaine)

• **Supprimer une sous-chaine uniquement au début d'une chaine :** TRIM(LEADING sous-chaine FROM chaine)

• **Supprimer une sous-chaine uniquement à la fin d'une chaine :** TRIM(TRAILING sous-chaine FROM chaine)

• **Récupérer une partie d'une chaine à partir d'une position :** SUBSTRING(chaine, position)

• **Récupérer une partie d'une chaine à partir d'une position et jusqu'à une autre position :** SUBSTRING(chaine, position1, position2)

• **Renvoyer la position de la première occurrence d'une sous-chaine dans une chaine :** INSTR(chaine, sous-chaine) **OU :** POSITION(sous-chaine IN chaine) **OU :** LOCATE(sous-chaine, chaine)

• **Renvoyer la position de la première occurrence d'une sous-chaine dans une chaine à partir d'une position :** LOCATE(sous-chaine, chaine, position)

• **Récupérer un nombre de caractère de la fin d'une chaine :** LEFT(chaine, nombre)

• **Récupérer un nombre de caractère du début d'une chaine :** RIGHT(chaine, nombre)

• **Inverser une chaine :** REVERSE(chaine)

• **Concaténer deux chaines :** CONCAT(chaine1, chaine2)

• **Insérer une chaine dans une autre chaine à partir d'une position pour une certaine longueur :** INSERT(chaine, position, longueur, autre_chaine)

• **Remplacer toutes les occurrences d'un caractère dans une chaine :** REPLACE(chaine, caractère_à_remplacer, caractère_qui_remplace)

• **Concaténer plusieurs chaines dans une seule :** CONCAT(chaine1, chaine2)

• **Concaténer plusieurs chaines dans une seule en ajoutant un séparateur entre les chaines :** CONCAT(séparateur, chaine1, chaine2)

• **Renvoyer l'index de la première chaine qui correspond à la chaine de recherche parmi une liste de chaine** (l'index commence à 1 et renvoie 0 si la chaine n'est pas trouvée) **:** FIELD(chaine_recherchée, chaine1, chaine2, chaineN)

• **Renvoyer le code ASCII du premier caractère d'une chaine :** ASCII(chaine)

13 FONCTIONS TEMPORELLES

• **Renvoyer la date et l'heure actuelles du système dans le format 'YYYY-MM-DD HH:MI:SS' :** NOW()

• **Renvoyer le jour du mois :** DAY(date)

• **Renvoyer le jour de la semaine** (entre 1 et 7) **:** DAYOFWEEK(date) (1 correspond au dimanche)

• **Renvoyer le jour de la semaine** (entre 0 et 6) **:** WEEKDAY(date) (0 correspond au lundi)

• **Renvoyer le nom du jour de la semaine en anglais :** DAYNAME(date)

• **Renvoyer le jour de l'année** (entre 1 et 366) **:** DAYOFYEAR(date)

• **Renvoyer le numéro de la semaine de l'année** (entre 0 et 53) **:** WEEK(date) **OU :** WEEKOFYEAR(date) (WEEK() peut renvoyer 0 pour les semaines qui chevauchent deux années différentes, alors que WEEKOFYEAR() renvoie toujours un numéro de semaine compris entre 1 et 53)

• **Renvoyer l'année correspondant au numéro de la semaine de l'année :** YEAROFWEEK(date)

• **Renvoyer le mois** (entre 1 et 12) **:** MONTH(date)

• **Renvoyer le nom du mois en anglais :** MONTHNAME(date)

• **Renvoyer l'année :** YEAR(date)

• **Renvoyer l'heure à partir d'une date et heure combinées :** TIME(datetime)

• **Renvoyer l'heure (entre 0 et 23) :** HOUR(heure/datetime)

• **Renvoyer les minutes (entre 0 et 59) :** MINUTES(heure/datetime)

• **Renvoyer les secondes (entre 0 et 59) :** SECOND(heure/datetime)

• **Formater une date dans une chaine :** DATE_FORMAT(date, 'chaine %Y-%m-%d')

Formats de date et heure pour la fonction DATE_FORMAT :

— %d : le jour du mois avec un zéro initial (01 à 31)

— %e : le jour du mois sans un zéro initial (1 à 31)

— %D : le jour du mois avec un suffixe (1er, 2ème, 3ème, etc.)

— %w : le jour de la semaine en chiffre (0 pour dimanche, 1 pour lundi, etc.)

— %W : le numéro de la semaine dans l'année (01 à 53)

— %a : l'abréviation du nom du jour de la semaine en anglais (Mon pour lundi, Tue pour mardi, etc.)

— %m : le mois en chiffre (01 pour janvier, 02 pour février, etc.)

— %c : le mois en chiffre sans zéro initial (1 à 12)

— %M : l'abréviation du nom du mois en anglais (JAN pour janvier, FEB pour février, etc.)

— %b : le nom abrégé du mois en français (janv. pour janvier, févr. pour février, etc.)

— %y : l'année sur deux chiffres (21 pour 2021, etc.)

— %Y : l'année sur quatre chiffres (2021, etc.)

— %r : l'heure, au format AM/PM, avec la date (Wed, 27 Mar 2021 02:45:57 PM, etc.)

— %t : une tabulation

— %h : l'heure en format 12 heures (01 à 12)

— %H : l'heure en format 24 heures (00 à 23)

— %I : l'heure en format 12 heures avec un zéro initial (01 à 12)

— %i : les minutes avec un zéro initial (00 à 59)

— %k : l'heure en format 24 heures sans zéro initial (0 à 23)

— %s : les secondes avec un zéro initial (00 à 59)

— %S : les secondes sans un zéro initial (0 à 59)

— %p : AM ou PM (pour les formats d'heure en 12 heures)

• **Renvoyer la différence en jours entre deux dates :** DATEDIFF(date1, date2)

• **Renvoyer la différence entre deux heures :** TIMEDIFF(heure1, heure2)

• **Renvoyer la différence entre deux dates en fonction de l'unité de temps spécifiée :** TIMESTAMPDIFF(unité, date1, date2) (unité de temps sous la forme : YEAR, MONTH, DAY, HOUR, MINUTE, SECOND)

• **Ajouter une durée (en années, mois, jours, heures, minutes ou secondes) à une date donnée :** DATE_ADD(date, INTERVAL durée unité_temps)

• **Soustraire une durée (en années, mois, jours, heures, minutes ou secondes) à une date donnée :** DATE_SUB(date, INTERVAL durée unité_temps)

Formats des unités de temps pour TIMESTAMPDIFF et DATE_ADD :

— SECOND : une seconde

— MINUTE : une minute

— HOUR : une heure

— DAY : un jour

— WEEK : une semaine

— MONTH : un mois

— YEAR : une année

— MINUTE_SECOND : minutes et secondes

— HOUR_MINUTE : heures et minutes

— HOUR_SECOND : heures et secondes

— DAY_MINUTE : jours et minutes

— DAY_HOUR : jours et heures

— YEAR_MONTH : années et mois

14 UNION

• Créer une table reprend deux tables l'une à la suite de l'autre en supprimant les doublons :

SELECT …

UNION DISTINCT

SELECT …

• Créer une table reprend deux tables l'une à la suite de l'autre en conservant les doublons :

SELECT …

UNION ALL

SELECT …

• Remarques :

— Les deux requêtes SELECT doivent retourner le même nombre de colonnes.

— Les colonnes doivent être de même type, sinon tout sera converti en chaine de caractères.

— Il faut respecter l'ordre des colonnes.

15 ESPACE DE STOCKAGE (TABLESPACES)

• Tablespace : espace de stockage, conteneur de données.

• Créer un tablespace et l'associer à un répertoire : CREATE TABLESPACE tablespace LOCATION 'chemin' ;

• Changer le nom d'un tablespace : ALTER TABLESPACE tablespace RENAME TO nouveau_nom;

• Changer le propriétaire d'un tablespace : ALTER TABLESPACE tablespace OWNER nouveau_propriétaire;

• Supprimer un tablespace (il doit être vide pour pouvoir être supprimé) **:** DROP TABLESPACE tablespace;

• Créer une base de données dans un tablespace : CREATE DATABASE bdd TABLESPACE = tablespace;

• Créer une table en spécifiant un tablespace : CREATE TABLE table (…) TABLESPACE tablespace;

16 ESPACES DE NOMS (SCHÉMAS)

• **Schéma :** espace de noms qui est utilisé pour organiser les noms des objets de base de données. Cela permet d'avoir plusieurs objets portant le même nom dans une même base de données.

• **Créer un schéma dont le propriétaire est un certain utilisateur :** CREATE SCHEMA schéma AUTHORIZATION utilisateur;

• **Supprimer un schéma vide :** DROP SCHEMA schéma;

• **Supprimer un schéma qui possède des objets :** DROP SCHEMA schéma CASCADE;

• **Accéder à un objet d'un schéma :** schéma.objet