SQL note

➔ **Interclassement (encodage) :**
Meilleur interclassement | insensible a la casse (la table la plus large) = utf8_general_ci
Pour verifier l'interclassement d'une BD :
Req> SHOW VARIABLES (verifier caractere_set_database)

➔ **Vocabulaires :**
Cle pri maire = un champ special qui permet d'identifier de facon unique une table (PRIMARY/A_I)
Cle etrangere =
Cle candidate =
unique = qui ne doit pas comporter des doublons
contrainte =
index = ajouter a un champ un effet positif au niveau de la structure, i-e ameliorer la perf d'une table (inconveniant: occupe un peu plus de place si le champ prend des centaines de valeur)
innoDB = Supports transactions, row-level locking, foreign keys and encryption for tables

~ CREATION (CREATE) ~

■ **Creation d'une base de donnees :**

```
Req {
    CREATE DATABASE <database_name>
    DEFAULT CHARACTER SET utf8
    DEFAULT COLLATE utf8_general_ci;
}
```

■ **Creation d'une table :**

```
Req {
    USE <database_name>
        CREATE TABLE <nom_table> (
            <col1> <type1>(taille) [contrainte],
            <col2> <type2>(taille) [contrainte],
            <col3> <type3>(taille) [contrainte],
             ...
            <colN> <typeN>(taille) [contrainte]
```

[PRIMARY KEY (liste_cle_primaire)],
[FOREIGN KEY (champ_cle_etr) REFERENCES <tableref> (champ_de_tableref)
[ON DELETE CASCADE]]
) ENGINE=InnoDB DEFAULT CHARSET=utf8 ;

note : nbr de colonne max dans une table = 254cols
}

~ MODIFICATION (ADD / CHANGE / MODIFY / RENAME / UPDATE) ~

- **Ajout d'une colonne (champ) :**
Req {
ALTER TABLE <table_name>
ADD <col_name> <type>(<taille>) [option] ;

exemple :

ALTER TABLE COMMANDE //on ajoute la contrainte d'integrite referentielle dans la table COMMANDE
ADD FOREIGN KEY (id_client) REFERENCES CLIENT(id_client);
}

- **Renommage/Modification d'une colonne (champ) :**
Req {
ALTER TABLE <table_name>
CHANGE/MODIFY <col_name> <new_col_name> <type> [option] ;

note : on peut utiliser MODIFY a la place de CHANGE mais ce dernier ne peut modifier que le type (proprietes) de la colonne mais pas son nom

exemple :

ALTER TABLE employe
MODIFY (sal number(10,2), fonction varchar2(30)); //sal et fonction existent déjà dans la table
}

- **Renommage d'une table:**
Req> RENAME TABLE <table_name> TO <new_table_name> ;

- **Modifier les donnees:**
Req> UPDATE <table_name> SET <col1>=<value>, <col2>=<value>, <col3>=<value>
WHERE <conditions>;

note : vous pouvez ajouter l'option LIMIT 1 pour securiser vos modifications

exemple :

```
UPDATE employe
SET sal = sal * 1.25 //augmentation de 25% du salaire de tous les informaticiens
WHERE fonction = 'informaticien';

UPDATE employe
SET sal = sal * 1.0; //augmentation de 10% du salaire de tous les employes

//modifier plusieurs colonnes en meme temps, en separant les affectations par une
virgule
UPDATE employe
SET deptno = 1,
sal = 2000
WHERE code_service = 13;
```

~ SUPPRESSION (DROP / DELETE / TRUNCATE) ~

- **Suppression d'une base de donnees (operation irreversible):**
  Req> DROP DATABASE <database_name> ;

- **Suppression d'une table, colonnes + lignes (operation irreversible):**
  Req> DROP TABLE <table_name> ;

- **Suppression d'une ligne par condition:**
  Req> DELETE FROM <table_name> WHERE <condition> [option] ;

  note : vous pouvez securiser la suppression en ajoutant LIMIT n (avec n le nombre de ligne) comme option pour s'assurer que la requette ne supprime que n ligne, a savoir que cette requette ne supprime pas l'A_I d'une cle primaire

  attention : vous risquez de supprimer toutes les lignes si vous ne mettez pas la condition

  exemple :

```
DELETE FROM employe
WHERE code_service = 3 AND function = 'Comptable'; //suppression de tous les
comptables du service 3

DELETE FROM employe
WHERE empno = 10; //suppression de l'employe numero 10
```

- **Vidage propre d'une table (operation irreversible):**
  Req> TRUNCATE TABLE <table_name> ;

  note : contrairement a la requette drop, celle-ci va suprrimer meme l'A_I d'une cle primaire de la table (reinitialisation complete des proprietes d'une table)

~ INSERTION (INSERT) ~

- **Insertion d'une/plusieurs colonnes:**
  Req (values) {
  INSERT INTO <table_name>
  (<ID>, <col1>, <col2>, <col3>, ... , <colN>) VALUES (NULL, <value1>, <value2>, <value3>, ... , <valueN>) ;

  note : il faut respecter l'orde des colonnes pour bien completer les valeurs, on peut mettre directement des valeurs (en ignorant les champs <col>)
  }

  Req (set) {
  INSERT INTO <table_name>
  SET <col3>=<value>, <col6>=<value> ;

  note : SET permet de cibler une colonne specifique
  }

~ SELECTIONNER (SELECT / COUNT / MIN / MAX / LIKE) ~

- **Recuperer des donnees:**
  Req> SELECT <col1>, <col2>, <col3> ... <colN> FROM <table_name> WHERE <condition> LIMIT <n> [ORDER BY <col_name> <key_order>];

  note :

  Elle permet d'indiquer quelles colonnes, ou quelles expressions doivent etre retournees par l'interrogation (SELECT est une requette frequement utilisEe)

  Il est toujours fiable de mettre un **LIMIT <n>** (avec n un entier naturel) pour eviter une surcharge de memoire

  ORDER BY (facultatif) permet a la requette de trier automatiquement l'organisation des donnees (ordre alphabetique/ordre croissant/decroissant) par rapport au nom d'une colonne qu'on a choisi comme <col_name>, alors que <key_order> prend 2 valeurs comme ASC (qui est par defaut) et DESC qui signifie descendant

  On peut utiliser **l'expression reguliere *** pour recuperer tout les champs

- **Compter des donnees:**
  Req> SELECT COUNT(*) as <var_name> FROM <table_name> [GROUP BY <col_name>]

  note :

  Le mot cle **'as'** permet de stocker la valeur recuperEe par **COUNT(*)** dans la variable **<var_name>** (quand vous utilisez PHP, il est plus facile de recuperer <var_name>)

  GROUP BY (facultatif) permet a la requette d'isoler (ignorer la fusion) le resultat a partir du <col_name>

  exemple :

  SELECT COUNT(*) as total, sexe as sexe FROM populations GROUP BY sexe //cette requette permet de recuperer le nombre total groupE par sexe (m/f) dans la table population

- **Recuperer la valeur MINimum et MAXimum des donnees:**
  Req> SELECT MIN/MAX(<col_name>) as <var_name> FROM <table_name> [ORDER BY <col_name> <key_order>] [GROUP BY <col_name>]

- **Recuperer une/plusieurs donnee(s) a partir d'un mot cle:**
  Req> SELECT * FROM <table_name> WHERE <col_name> LIKE "[%]<key>[%]"

  note :

  LIKE permet de retourner des donnees specifiques a partir du mot cle <key>

  Le **joker %** signifie n'importe quel valeur

  exemple :

  SELECT * FROM eleves WHERE nom LIKE "R%" AND classe = "Seconde" //cette requette permet de recuperer tout les eleves dont le nom commence par R et qui se trouvent dans la classe de seconde

  SELECT * FROM eleves WHERE pnom LIKE "%o%" AND age < 18 //cette requette permet de recuperer tout les eleves dont le prenom contient 'o' et qui sont mineures

~ FONTIONS ~

INTEGER

- **ABS(value) =** valeur absolu
- **ROUND(value)** = arrondiser une valeur
- **CELL(value)** = arrondi supperieur | ex : CELL(1.2) donne 2
- **FLOOR(value)** = arrondi inferieur | ex : FLOOR(1.2) donne 1
- **SQRT(value)** = racine carre d'une valeur
- **TRUNCATE(value)** = nombre de chiffre apres virgule a recuperer | ex : TRUNCATE(3.14159265359,2) donne 3.14
- **MOD(value)** = reste de la division entre 2 valeurs | ex : MOD(10,2) donne 0

CHAINES

- **CONCAT =** concatenation (2/plusieurs chaines de caracteres) | ex : SELECT * CONCAT (nom, ' ' , pnom) as fullname FROM users //cette requette permet de concatener le nom et le prenom de l'utilisateur pour donner son nom complet dans la variable fullname

~ OTHERS ~

- **SQL code request (une requette speciale)**

```
Req {
    USE <database_name>
        SHOW CREATE TABLE <table_name> ;
}
```

- **Date format (american) =** Y/M/D h:m:s
- Il est preferable que le nom d'une **table/colonne** soit entoure d'une backquote `...` et la valeur par une double quote "..."
- On n'a pas besoin de **mettre de guillemet** pour les entres des **entiers**
- La **deuxieme notation** de l'option LIMIT est comme suit : LIMIT a, b (avec a et b sont des entiers naturels) dont a signifie le point de depart et b la derniere lignes a retourner
- la valeur **NULL** est considerEe comme une absence de valeur (pas un zero ou une chaine vide) | tandis que **NOT NULL** signifie que le champ est obligatoire
- Les **operateurs de comparaison** dans SQL sont : egal(=), different(!=|<>), supperieur|egal(>|=), inferieur|egal(<|=)
- **WHERE** age = 18 **OR** age = 19 **OR** age = 20 peut se traduire par **WHERE** age **IN** (18,19,20)

~ EXEMPLES ~

Creation d’une table + colonnes

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	id	int(11)			Non	*Aucun(e)*		AUTO_INCREMENT
2	titre	varchar(255)	utf8_general_ci		Non	*Aucun(e)*		
3	slug	varchar(60)	utf8_general_ci		Non	*Aucun(e)*		
4	contenu	longtext	utf8_general_ci		Non	*Aucun(e)*		
5	date_creation	datetime			Non	*Aucun(e)*		
6	date_modification	timestamp			Non	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()

```
CREATE TABLE `articles` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `titre` varchar(255) NOT NULL,
 `slug` varchar(60) NOT NULL,
 `contenu` longtext NOT NULL,
 `date_creation` datetime NOT NULL,
 `date_modification` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),
 PRIMARY KEY (`id`),
 UNIQUE KEY `url` (`slug`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Modification d’une colonne

Nom	Type	Taille/Valeurs*	Valeur par défaut	Interclassement	Attributs
sexe Choisir à partir des colonnes centrales	CHAR	1	Tel que défini : h	utf8_general_ci	

Aperçu SQL

```
ALTER TABLE `users` CHANGE `sexe` `sexe` CHAR(1) CHARACTER SET
utf8 COLLATE utf8_general_ci NOT NULL DEFAULT 'h';
```