Le langage SQL

Fondamentaux du langage SQL

Qu’est ce que le SQL ?

- **S**tructured **Q**uery **L**anguage = Langage d'interrogation structuré
- Permet de consulter une base de données (requêtes sélection) et aussi de créer, modifier, supprimer tables et enregistrements
- Langage normalisé mais implémentations parfois différentes (dialectes)

Rappel : la référence du langage MySQL est ici
http://dev.mysql.com/doc/refman/5.7/en/

Qu’est ce que le SQL ?

- L4G (langage de quatrième génération)
- Langage déclaratif, non procédural (procédural : C, Pascal...)
- Inspiré du modèle d'algèbre relationnel de E.F. Codd (1970)
- Utilisé sur Oracle, MySQL, PostGreSQL, Access…

Historique

- 1970 Développement de langages de requête
- 1976 Sequel (IBM)
- 1982 Sequel → SQL
- 1986 SQL1 (SQL-86) → norme ANSI
- 1987 SQL1 → norme ISO
- 1989 SQL1 révisé (SQL-89) → norme ANSI/ISO
- 1992 SQL2 (SQL-92) → norme ANSI/ISO
- 1999 SQL-99 → norme ANSI/ISO
- 2003 SQL:2003 → norme ANSI/ISO
- 2008 SQL:2008 → norme ANSI/ISO
- 2011 SQL:2011 → norme ANSI/ISO
- 2013 SQL:2013 → version SQL 5.7

SQL : un langage de programmation

- C’est quoi un langage de programmation ?
  - Lien avec la définition des BDD
- Attention à la syntaxe
- Langage déclaratif :
  - On manipule directement les données
  - Il y a des “mots clés”

Syntaxe du SQL

- Séparateur d'instructions : point virgule
- Séparateur de mots-clés :
  - espace, tab, fin de ligne
- Mots-clés du SQL non sensibles à la casse
  - Convention : mots-clés en majuscules
- Noms d'objets (table, champ…) sensibles à la casse

Syntaxe du SQL

- Eviter les mots réservés pour les noms d'objets :
  - action, as, date, from, is, section, session, table, union, work, zone…
- Entourer de '…' ou "…" les noms d'objets comportant un espace
  - ex : 'moyenne des étudiants'
  - Valeurs de chaînes et date entre ' ' ou " " dans les expressions

Syntaxe du SQL

- Des commentaires sont possibles sous plusieurs formes :
  - /* commentaire sur plusieurs lignes */
  - -- espace puis commentaire sur une ligne
  - # commentaire sur une ligne

Syntaxe du SQL

- Rappel :
  - pour séparer les instructions on place ‘;’
  - souvent en fin de ligne
- Pour les chaines de caractères
  - \n : retour chariot
  - \t : tabulation
  - \’ : l’apostrophe
  - \ est le caractère d’échappement
  - \\ : c’est \
- Attention pour ‘ on peut faire ‘’

Ecrire du SQL

- A éviter
  - SELECT licence, MIN(note), MAX(note),AVG(note) AS Moyenne FROM etud WHERE licence=1 GROUP BY licence ORDER BY Moyenne ;
- A privilégier : saut de ligne, indentation
  - SELECT licence,
    MIN(note), MAX(note),
    AVG(note) AS Moyenne
    FROM etud
    WHERE licence=1
    GROUP BY licence
    ORDER BY Moyenne ;

L’instruction SELECT

- Comportement particulier :
  - évalue l’expression qui suit
  - sert pour des calculs
- Exemple :
  - Select 3*(6-2)
  - Select “coucou”

Exemple de table en SQL

num	nom	prenom	dateN	note	dep	licence
1	Martin	Véra	85-10-31	13.5	77	2
2	Martin	Annie	85-12-31	15.5	75	1
3	Dupont	Sylvie	83-02-03	15.0	77	2
4	Martin	Annie	83-10-22	05.7	93	1
5	Dupond	Laurent			92	2
6	Lefèvre	Laurent		11.5		3

Les données dans SQL

La “valeur” NULL

- Pour une valeur inconnue
  - ex : note, date de naissance…
- Pour une valeur non pertinente
  - ex : nom de jeune fille pour un garçon
- NULL ≠ zéro, NULL ≠ chaîne vide
  - ex : note NULL (inconnue) est différent de la note 0
- Null ne se traite pas comme une autre valeur
  - … = NULL (uniquement pour l'affectation)
  - … IS NULL (pour des comparaisons)
  - … IS NOT NULL

Rappel des différents types de données

- Numérique
  - Entier, flotants
  - Attention aux valeurs approchées
- Texte
  - Chaine de caractère, textes longs
- Date
  - Heure, Date, Mois …
- Enumérations
  - Homme-Femme, Pique-Carreau-Coeur-Trefle...

Types de données numériques en SQL

- Entier signé (+,-) ou non signé (+)
  - TINYINT, BIT, BOLLEAN, BOOL
  - SMALLINT
  - MEDIUMINT
  - INT (=INTEGER)
  - BIGINT
- Réel (Nombre à virgule flottante)
  - DEC (=DECIMAL)
  - FLOAT
  - DOUBLE (=REAL)

Représentation EXACTE

Représentation APPROCHEE

Données MySQL 5 : entiers

Désignation	octets	valeurs	précision	de	à	soit
TINYINT	1	2^8	signed unsigned	-128 0	127 255	(2^7)-1 (2^8)-1
SMALLINT	2	2^{16}	signed unsigned	-32 768 0	32 767 65 535	(2^{15})-1 (2^{16})-1
MEDIUMINT	3	2^{24}	signed unsigned	-8 388 608 0	8 388 607 16 777 215	(2^{23})-1 (2^{24})-1
INT ou INTEGER	4	2^{32}	signed unsigned	-2 147 483 648 0	2 147 483 647 4 294 967 295	(2^{31})-1 (2^{32})-1
BIGINT	8	2^{64}	signed unsigned	-9 223 372 036 854 775 808 0	9 223 372 036 854 775 807 18 446 744 073 709 551 615	(2^{63}) -1 (2^{64}) -1

UNSIGNED : option à préciser. Par défaut, les nombres sont signés (signed)
ZEROFILL : ajoute les zéros à gauche pour unsigned (positif). ex : zerofill int(4)
Pas de booléen : utiliser un CHAR(1) ou TINYINT(1) ou ENUM

Données MySQL 5 : réels

Désignation	octets	signe	de	à
FLOAT	4	signed unsigned	-3.402823466E+38 0	+3.402823466E+38 +3.402823466E+38
REAL ou DOUBLE	8	signed unsigned	-1.7976931348623157E+308 0	1.797693134862357E+308 1.797693134862357E+308
DECIMAL(M,D) ou DEC(M,D)	M + 1 ou 2	signed unsigned	-1.7976931348623157E+308 0	-1.797693134862357E+308 1.797693134862357E+308

ZEROFILL : option pour mettre des zéro non significatifs à gauche.
DECIMAL : enregistré comme une chaîne ; adapté aux données monétaires.
M : nombre total de chiffres (avant et après la virgule)
D : nombre de décimales (chiffres après la virgule)

Types de données chaîne en SQL

- Chaîne (caractères)
  - CHAR (=CHARACTER)
  - VARCHAR (=CHARACTER VARYING)
  - Liste de valeurs : ENUM, SET
  - TEXT : TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT
  - BLOB : TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB

Données MySQL 5 : chaîne et enum.

Chaînes fixes et variables				
Désignation	octets	Longueur	nombre de caractères	optionnel
CHAR(n)	n	fixe de n	1 ≤ n ≤ 255	BINARY
VARCHAR(n)	Lg+1	n max Lg variable	1 ≤ n ≤ 255	BINARY

Enumération et ensemble de valeurs				
Désignation	octets	Liste de valeurs	choix de valeurs	nombre max. de valeurs
ENUM	1 ou 2	'val1','val2'	une	65 635
SET	1,2,3,4 ou 8	'val1','val2'	plusieurs	64

CHAR(n) : traitement plus rapide
VARCHAR(n) : optimisation de la taille de la base de données
Option BINARY : comparaisons et tris sensibles à la casse (maj, min, accents)

Données MySQL 5 : blob et textes

Blob et Text				
Désignation	octets	valeurs	longueur maximale de la chaîne	soit
TINYBLOB TINYTEXT	1	2^8	255	(2^8) -1
BLOB TEXT	2	2^16	65 535	(2^16) -1
MEDIUMBLOB MEDIUMTEXT	3	2^24	16 777 215	(2^24) -1
LONGBLOB LONGTEXT	4	2^32	4 294 967 295	(2^32) -1

Stockage de volume important de données
TEXT : objet texte (insensible à la casse)
BLOB : Binary Large OBject = objet binaire (image,….) ressemble à un TEXT mais BINARY

Type de données date en SQL

- Ce sont les données temporelles
  - DATE, YEAR
  - TIME, TIMESTAMP
- Affiche 0 quand date invalide
  - hors intervalle, mois > 12, jour > 31 …
- Format de saisie acceptés :
  - Chaîne avec délimiteurs divers
    - chaîne ou nombre sans délimiteurs sur un nombre de position valide (6, 8, 10, 12, 14)
    - Ex : 'YYYYMMDDHHMMSS', 'YYYY-MM-DD HH:MM:SS', 'YYYY-MM-DD-HH-MM-SS', 'YYYY/MM/DD HH-MM-SS',

Données MySQL 5 : date

Désignation	octet	format	de	à
DATE	3	YYYY-MM-DD	0001-01-01	9999-12-31
YEAR(4)	1	YYYY	1901	2155
YEAR(2)	1	YY	70 (1970)	69 (2069)
TIME	3	HHH:MM:SS	-838:59:59	838:59:59
DATETIME	8	YYYY-MM-DD HH:MM:SS	0001-01-01 00:00:00	9999-12-31 23:59:59
TIMESTAMP TIMESTAMP(14)	4	YYYY-MM-DD HH:MM:SS YYYYMMDDHHMMSS	1970-01-01 00:00:00 19700101000000	2037-12-31 23:59:59 20371231235959
TIMESTAMP(12)	4	YYYYMMDDHHMM	1970-01-01 00:00	2037-12-31 23:59
TIMESTAMP(10)	4	YYYYMMDDHH	1970-01-01 00	2037-12-31 23
TIMESTAMP(8)	4	YYYYMMDD	1970-01-01	2037-12-31
TIMESTAMP(6)	4	YYMMDD	70-01-01	37-12-31
TIMESTAMP(4)	4	YYMM	70-01	37-12
TIMESTAMP(2)	4	YY	70	37

TIMESTAMP prend la date/heure courante (pour transaction).

Exemples & problèmes

Type de données	Valeur entrée	Valeur stockée
DATE	2004-12-00	2004-12-00 (date valide, jour inconnu)
DATE	2004-13-31	0000-00-00 (date invalide, mois 0 à 12) ou erreur
DATE	2004-12-32	0000-00-00 (date invalide, jour 0 à 31)
DATE	2004-02-31	Peut être accepté en général produit une erreur
INT(5)	12,567	13 (arrondi)
TINYINT(1)	1234.567	127 (le plus grand) ou erreur à la génération
DOUBLE(3,1) / FLOAT (3,1) / DEC(3,1)	12.59	12.6 (arrondi)

Les fonctions sur les données

Les fonctions SQL

- Existence de nombreuses fonctions sur les données elles-mêmes :
  - Fonction arithmétique simples
  - Fonction mathématique complexes
  - Fonction logiques
  - Fonction comparaison
  - Fonction comparaison chaine
  - Fonction de chaine
  - Fonction de date

Opérateurs mathématiques et logiques

Name	Description
`DIV`	Integer division
`/`	Division operator
`-`	Minus operator
`%`, `MOD`	Modulo operator
`+`	Addition operator
`*`	Multiplication operator
`-`	Change the sign of the argument

Name	Description
`AND`, `&&`	Logical AND
`NOT`, `!`	Negates value
`||`, `OR`	Logical OR
`XOR`	Logical XOR

Ordre de priorité des opérateurs

NOT

- (négatif)

^

* / % MOD

+ -

= <> != > < >= <= IS LIKE REGEXP IN

BETWEEN

AND &&

OR ||

Fonction mathématiques complexes

Name	Description
ABS()	Return the absolute value
ACOS()	Return the arc cosine
ASIN()	Return the arc sine
ATAN()	Return the arc tangent
ATAN()	Return the arc tangent of the two arguments
CEIL()	Return the smallest integer value not less than the argument
CEILING()	Return the smallest integer value not less than the argument
CONV()	Convert numbers between different number bases
COS()	Return the cosine
COT()	Return the cotangent
CRC32()	Compute a cyclic redundancy check value
DEGREES()	Convert radians to degrees
EXP()	Raise to the power of
FLOOR()	Return the largest integer value not greater than the argument
LN()	Return the natural logarithm of the argument
LOG()	Return the natural logarithm of the first argument

Name	Description
LOG10()	Return the base-10 logarithm of the argument
LOG2()	Return the base-2 logarithm of the argument
MOD()	Return the remainder
PI()	Return the value of pi
POW()	Return the argument raised to the specified power
POWER()	Return the argument raised to the specified power
RADIANS()	Return argument converted to radians
RAND()	Return a random floating-point value
ROUND()	Round the argument
SIGN()	Return the sign of the argument
SIN()	Return the sine of the argument
SQRT()	Return the square root of the argument
TAN()	Return the tangent of the argument
TRUNCATE()	Truncate to specified number of decimal places

Fonctions de comparaison

Name	Description
`BETWEEN ... AND ...`	Check whether a value is within a range of values
`COALESCE()`	Return the first non-NULL argument
`=`	Equal operator
`<=>`	NULL-safe equal to operator
`>`	Greater than operator
`>=`	Greater than or equal operator
`GREATEST()`	Return the largest argument
`IN()`	Check whether a value is within a set of values
`INTERVAL()`	Return the index of the argument that is less than the first argument
`IS`	Test a value against a boolean
`IS NOT`	Test a value against a boolean
`IS NOT NULL`	NOT NULL value test

Name	Description
`IS NOT NULL`	NOT NULL value test
`IS NULL`	NULL value test
`ISNULL()`	Test whether the argument is NULL
`LEAST()`	Return the smallest argument
`<`	Less than operator
`<=`	Less than or equal operator
`NOT BETWEEN ... AND ...`	Check whether a value is not within a range of values
`!=, <>`	Not equal operator
`NOT IN()`	Check whether a value is not within a set of values

Fonction de comparaison de chaine

Name	Description
`LIKE`	Simple pattern matching
`NOT LIKE`	Negation of simple pattern matching
`STRCMP()`	Compare two strings

LIKE / NOT LIKE : utilise le “pattern matching” avec des caractères spéciaux

% : remplace de 0 à n caractères (n’importe lesquels)

_ : remplace exactement 1 caractère (n’importe lequel)

Ex : ‘image.png’ LIKE ‘%.png’ renvoie TRUE (1)

Attention n’est pas case sensitive (‘abc’ LIKE ‘ABC’ est vrai)

STRCMP : Prend 2 argument. Renvoie -1 si la première chaine est plus petite que la seconde dans l’ordre lexicographique, 1 si elle est plus grande et 0 si elles sont égales.

Fonctions de chaines

- Beaucoup (beaucoup) de fonctions
- Notamment :
  - Passage en majuscule / minuscule : LOWER / UPPER
  - Concatener 2 chaines : CONCAT
  - Enlever les espaces : LTRIM / RTRIM
  - Renvoyer une sous chaine : SUBSTRING

Fonction de dates

- Beaucoup de fonctions dont :
  - Calcul sur les dates avec des intervales :
    - DATESUB, DATEADD, TIMEDIFF ...
  - Conversion de date :
    - DATE, YEAR, MINUTE...
  - Récupération de la date courante :
    - CURDATE, DAYOFMONTH...

Récupération de date/temps

CURRENT_DATE CURRENT_DATE() CURDATE() AAAA-MM-JJ
CURRENT_TIME CURRENT_TIME() CURTIME() HH-MM-SS
NOW() AAAA-MM-JJ HH-MM-SS
YEAR(date) année (4 chiffres)
MONTH(date) mois (chiffre)
DAYOFMONTH() 26 (nème jour du mois)
DAYOFYEAR() 310 (nème jour de l'année)
EXTRACT(exp FROM date) exp=month, year…
DATE_FORMAT(date, format) date avec formatage demandé
TO_DAYS(date) nb jours écoulés entre date et an 0
FROM_DAYS(nb) inverse de To_DAYS(), retourne la date
UNIX_TIMESTAMP(date) nb jours écoulés entre date et 1970-01-01

Formats pour DATE_FORMAT()

- Syntaxe : DATE_FORMAT(date,'format')

%y an (2 chiffres) : 97
%m mois (01 à 12) : 04

%d jour (01 à 31) : 08
%a jour (court) : Sat

%j N° jour (1 à 365)
%U semaine de l'année
%r heure (12h) : 11:59:30 PM
%I heure (12h) : 11
%p AM ou PM
%S secondes (2 chiffres) : 53

%Y an (4 chiffres) : 1997
%b mois (court) : Apr
%M mois (long) : April
%e jour (1 à 31) : 8
%W jour (long) : Saturday
%w jour de semaine (Di=0, Lu=6)

%u idem (début=lu)
%T heure (24h) : 23:59:30
%H heure (24h) : 23
%i minutes (2 chiffres) : 51

Fonction de controle (test)

- IF(condition,val_vrai,val_faux)
  - ex : IF(note>=10,'admis','refusé')
- IFNULL(valeur,val_faux)
  - ex : IFNULL(note,'abs')
- Test avec cas
  - CASE val

    WHEN comp1 THEN res1

    WHEN comp2 THEN res2

    ELSE res3

    END

Les commandes SQL

Principales commandes du SQL

- Sélection de champs/lignes : SELECT …
  - FROM …
  - WHERE …;
- Suppression de lignes : DELETE
  - FROM …
  - WHERE …;
- Mise à jour de données : UPDATE …
  - SET …
  - WHERE …;
- Insertion de lignes : INSERT INTO …
  - VALUES ...;
- Création de table : CREATE …
- Suppression de table : DROP …
- Modification de structure : ALTER ….

Classification des commandes SQL

SQL				
interactif			intégré	dynamique
LDD (DDL) Définition	**LMD (DML)** Manipulation	**LCD (DCL)** Contrôle		
CREATE DROP ALTER	INSERT DELETE UPDATE Interrogation SELECT	GRANT REVOKE CONNECT COMMIT ROLLBACK SET	DECLARE CURSOR FETCH	PREPARE DESCRIBE EXECUTE

L= langage (language) **D= Donnée (Data)**

Commande de création de base/table

Création et sélection de base

- Create de base : CREATE DATABASE
  - Ex : CREATE DATABASE Zoo;
- Sélection d'une base : USE
  - USE Zoo;
- Utile : création (ou suppression) en fonction de si cela existe : IF EXISTS / IF NOT EXISTS
  - Ex : CREATE DATABASE IF NOT EXISTS Zoo;

Suppression de base

- Attention c’est irreversible
  - Mais il y a quand même des outils …
- Suppression d’une base : DROP
  - Ex : DROP DATABASE Zoo;
- Fonctionne aussi avec les tables :
  - DROP TABLE IF EXISTS Animaux;

Création de tables

- Syntaxe :

```
CREATE TABLE nom_table {
   nom_chp1 type modifieurs,
   nom_chp2 type modifieurs,
   ...
};
```

```
CREATE TABLE etud (
    num INT UNSIGNED NOT NULL PRIMARY KEY,
    nom VARCHAR(30) NOT NULL,
    prenom VARCHAR(25) NOT NULL,
    dateN DATE,
    note DECIMAL(3,1) UNSIGNED ZEROFILL,
    dep CHAR(2),
    licence TINYINT(1)
);
```

Création de table

- Les modifieurs importants :
  - NULL / NOT NULL
  - PRIMARY KEY
  - AUTO INCREMENT
  - UNSIGNED : à mettre juste après le type
  - ZEROFILL
  - DEFAULT xxx : Fixe la valeur par défaut

Modification d’une table

- Commande : ALTER TABLE
- Beaucoup de possibilités :
  - Supprimer une colonne
    - ALTER TABLE Animaux DROP COLUMN Pseudo
  - Ajouter une colonne
    - ALTER TABLE Animaux ADD COLUMN Pseudo VARCHAR(30);
  - Modifier une colonne
    - ALTER TABLE Animaux MODIFY COLUMN a TINYINT NOT NULL;
  - Renommer une table
    - ALTER TABLE Animaux RENAME Mamiferes

Insertion des données

- Syntaxe 1 :

```
INSERT INTO nom_table
VALUES (val1, val2, val3 … );
```

- Exemple :

```
INSERT INTO etud
VALUES (7,'Martin','Véra' ,'1988-01-01',13.5,75,1) ;
```

Insertion des données

- Syntaxe 2 :

```
INSERT INTO nom_table (chpX, chpY, chpZ...)
VALUES (valX, valY, valZ);
```

- Exemple :

L'ordre n'a pas d' importance

```
INSERT INTO etud (licence, nom, prenom)
VALUES (1,'Martin','Véra') ;
```

Insertion de données en masse

- Instruction LOAD DATA
- A faire (TD1)

Suppression de données

- Attention : peut avoir un impact sur des tables liées (par des clés étrangères par exemple)
  - Syntaxe générale

    ```
    DELETE FROM nom_table
        WHERE condition ;
    ```
  - Suppression de toutes les lignes de la table (-> table vide) !

    ```
    DELETE FROM etud ;
    ```
  - Suppression des lignes d'étudiants de licence 3

    ```
    DELETE FROM etud
        WHERE licence=3 ;
    ```
  - Suppression des lignes d'étudiants nommés Jean Dupont

    ```
    DELETE FROM etud
        WHERE nom='Dupont' AND prenom='Jean' ;
    ```

Mise à jour des données

- Changer une note :

```
UPDATE etud SET note=18 WHERE  num=6;
```

- Changer le prénom :

```
UPDATE etud SET prenom='Véro'
WHERE nom='Martin' AND prenom='Véra' ;
```

- Changer le prénom et le département :

```
UPDATE etud SET prenom='Véro', dep=NULL
WHERE num=4 ;
```

- Augmenter toutes les notes de 2 :

```
UPDATE etud SET note=note+2 ;
```

Interroger une BD : la sélection

Interroger une BD : SELECT

- SELECT permet d'interroger une base de données
- Le résultat est une nouvelle table (temporaire)
- Sélection de colonnes (projection) dans l'ordre indiqué
- Sélection de lignes (restriction / sélection)
- Des clauses supplémentaires permettent de :
  - faire des calculs sur des regroupements de lignes
  - trier les lignes
  - afficher un certain nombre de lignes de résultats

Syntaxe du SELECT

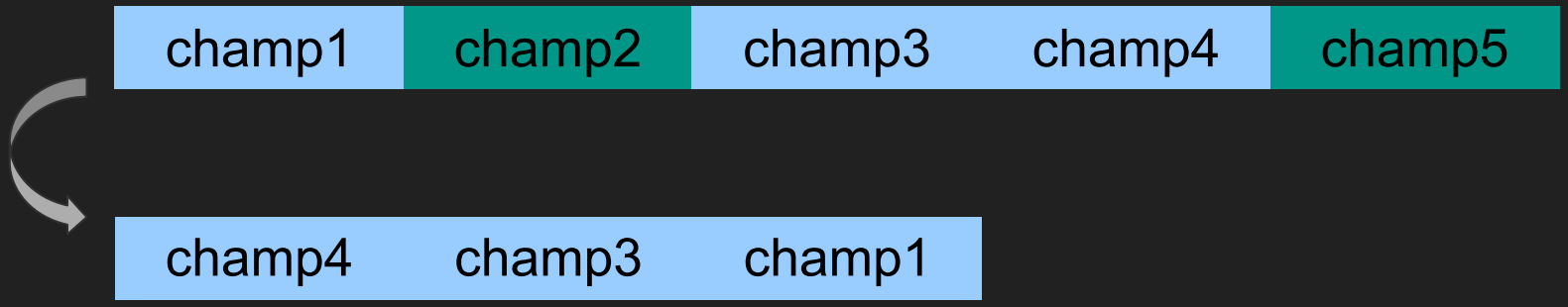
```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
    select_expr [, select_expr ...]
    [FROM table_references
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}
        [ASC | DESC], ...]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
        [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

Syntaxe minimale

SELECT … FROM table;

Quelques exemples

- SELECT * FROM table1;
  - On obtient toute la table
- SELECT champ3 FROM table1;
  - On obtient uniquement le champ3 de la table
- SELECT champ4, champ3, champ1 FROM table1;



Sélection de plusieurs colonnes

- Note, nom et prénom de tous les étudiants
  SELECT note, nom, prenom FROM etud;

note	nom	prenom
13.5	Martin	Véra
15.5	Martin	Annie
15.0	Dupont	Sylvie
05.7	Martin	Annie
	Dupond	Laurent
11.5	Lefèvre	Laurent

Opérateur * : toutes les colonnes

- Liste deFROM toutes les colonnes de la table

  SELECT * etud;

- Identique à :

  SELECT num, nom, prenom, date_n, note, dep, licence FROM etud;

num	nom	prenom	dateN	note	dep	licence
1	Martin	Véra	85-10-31	13.5	77	2
2	Martin	Annie	85-12-31	15.5	75	1
3	Dupont	Sylvie	83-02-03	15.0	77	2
4	Martin	Annie	83-10-22	05.7	93	1
5	Dupond	Laurent			92	2
6	Lefèvre	Laurent		11.5		3

Alias des noms de colonne

- En utilisant AS

```
SELECT  prenom AS Prénom,
        nom AS 'Nom de famille'
FROM etud;
```

- Sans utiliser AS (facultatif)

```
SELECT  prenom Prénom,
        nom 'Nom de famille'
FROM etud;
```

Prénom	Nom de famille
...	...

Gérer les doublons : ALL & DISTINCT

SELECT *
FROM etud ;

Nom	prenom	...	dep
Martin	Véra	...	77
Martin	Annie	...	75
Dupont	Sylvie	...	92
Martin	Annie	...	77
Dupond	Laurent	...	75
Lefèvre	Laurent	...	null
...	...	...	93
...	...	...	77
...	...	...	92
...	...	...	null

SELECT dep
FROM etud ;

dep
77
75
92
77
75
null
93
77
92
null

SELECT DISTINCT dep
FROM etud ;

dep
77
75
92
null
93

ALL est facultatif

Gérer les doublons

DISTINCT s'applique par ligne

SELECT ALL dep,licence
FROM etud;

dep	licence
93	2
75	1
75	2
77	1
75	3
75	2
null	3
93	2
93	2

SELECT DISTINCT dep,licence
FROM etud;

dep	licence
93	2
75	1
75	2
77	1
75	3
75	2
null	3
93	2
93	2

Gérer les doublons

DISTINCT s'applique par ligne

SELECT ALL dep,licence
FROM etud;

dep	licence
93	2
75	1
75	2
77	1
75	3
75	2
null	3
93	2
93	2

SELECT DISTINCT dep,licence
FROM etud;

dep	licence
93	2
75	1
75	2
77	1
75	3
null	3

Tri par ordre croissant (ascendant)

SELECT * FROM etud ORDER BY note;
SELECT * FROM etud ORDER BY note ASC;

num	nom	prenom	dateN	note	dep	licence
5	Dupond	Laurent			92	2
4	Martin	Annie	83-10-22	05.7	93	1
6	Lefèvre	Laurent		11.5		3
1	Martin	Véra	85-10-31	13.5	77	2
3	Dupont	Sylvie	83-02-03	15.0	77	2
2	Martin	Annie	85-12-31	15.5	75	1

Tri par ordre décroissant

SELECT * FROM etud ORDER BY note DESC;

num	nom	prenom	dateN	note	dep	licence
2	Martin	Annie	85-12-31	15.5	75	1
4	Dupont	Sylvie	83-02-03	15.0	77	2
1	Martin	Véra	85-10-31	13.5	77	2
6	Lefèvre	Laurent		11.5		3
4	Martin	Annie	83-10-22	05.7	93	1
5	Dupond	Laurent			92	2

Tri multiple

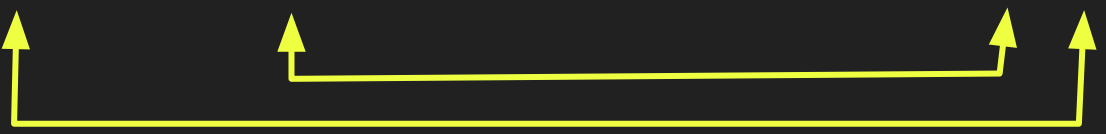
```
SELECT * FROM etud
    ORDER BY nom, prenom, dateN DESC;
```

num	nom	prenom	dateN	note	dep	licence
5	Dupond	Laurent			92	2
3	Dupont	Sylvie	83-02-03	15.0	77	2
6	Lefèvre	Laurent		11.5		3
2	Martin	Annie	85-12-31	15.5	75	1
4	Martin	Annie	83-10-22	05.7	93	1
1	Martin	Véra	85-10-31	13.5	77	2

Autres exemples de tri

Tri numérique, alphabétique ou chronologique

- SELECT * FROM t1 ORDER BY note;
- SELECT * FROM t1 ORDER BY note ASC;
- SELECT * FROM t1 ORDER BY note DESC;
- SELECT * FROM t1 ORDER BY nom, prenom,note;
- SELECT * FROM t1 ORDER BY 1, 2, 4;
- SELECT * FROM t1 ORDER BY note DESC, nom;
- SELECT nom, note, dep FROM t1 ORDER BY 3, 1;



Limiter le nombre de résultat

SELECT * FROM etud ORDER BY note DESC LIMIT 2;
équivalent à :
SELECT * FROM etud ORDER BY note DESC LIMIT 0,2 ;

nom	prenom	dateN	note	dep	licence
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	15.0	77	2
Martin	Véra	85-10-31	13.5	77	2
Lefèvre	Laurent		11.5		3
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2

Limiter le nombre de résultat

SELECT * FROM etud ORDER BY note DESC LIMIT 2;
équivalent à :
SELECT * FROM etud ORDER BY note DESC LIMIT 0,2 ;

nom	prenom	dateN	note	dep	licence
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	15.0	77	2

Limiter le nombre de résultat

SELECT * FROM etud LIMIT i,n ;
SELECT * FROM etud ORDER BY note DESC LIMIT 3,2;
Attention : Les lignes sont notées à partir de i=0 et non de i=1 !

i	rg
0	1
1	2
2	3
3	4
4	5
5	6

nom	prenom	dateN	note	dep	licence
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	15.0	77	2
Martin	Véra	85-10-31	13.5	77	2
Lefèvre	Laurent		11.5		3
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2

Ajouter des conditions : WHERE

- Syntaxe :

  SELECT … FROM … WHERE …;
- SELECT :
  - sélection de colonnes (projection)
- WHERE :
  - sélection de lignes (sélection = restriction)

Exemples de sélection

```
SELECT * FROM t1 WHERE nom='Martin';
SELECT * FROM t1 WHERE note >=10;
SELECT * FROM t1 WHERE note >10 AND dep=77;
```

champ1	champ2	champ3	champ4	champ5

Sélection avec comparaison de valeurs

Quels sont les étudiants dont le nom est Martin ?
SELECT * FROM etud WHERE nom='Martin' ;

nom	prenom	dateN	note	dep	licence
Martin	Véra	85-10-31	13.5	77	2
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	15.0	77	2
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2
Lefèvre	Laurent		11.5		3

Sélection avec comparaison de valeurs

Quels sont les étudiants dont le nom est Martin ?
SELECT * FROM etud WHERE nom='Martin' ;

nom	prenom	dateN	note	dep	licence
Martin	Véra	85-10-31	13.5	77	2
Martin	Annie	85-12-31	15.5	75	1
Martin	Annie	83-10-22	05.7	93	1

Rappel : les expressions constantes

- Chaîne : entre guillemets simples ' ' ou doubles " "
- Date : valeur entre guillemets en présence de séparateur - sans séparateur : guillemets facultatifs
- Echappement de l'apostrophe dans la valeur

Nombre

num=3
note=13.5
num='3'
note='13.5'

Chaîne

nom='Martin'
nom='Martin'
nom="Martin"
nom='L''Hote'
nom='L\'Hote'
nom="L'Hote"

Date avec séparateur (comme une chaîne)

date_n='1980-12-31'
date_n='1980/12/31'

Date sans séparateur

date_n='19801231'
date_n=19801231

Sélection avec connecteur logique

Liste des étudiants de licence 2 dont le nom est Martin

```
SELECT * FROM etud
    WHERE nom='Martin' AND licence = 2 ;
```

nom	prenom	dateN	note	dep	licence
Martin	Véra	85-10-31	13.5	77	2
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	15.0	77	2
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2
Lefèvre	Laurent		11.5		3

Sélection avec connecteur logique

Liste des étudiants de nom Dupont ou Dupond

```
SELECT * FROM etud
    WHERE nom='Dupont' OR nom='Dupond'  ;
```

nom	prenom	dateN	note	dep	licence
Martin	Véra	85-10-31	13.5	77	2
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	15.0	77	2
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2
Lefèvre	Laurent		11.5		3

Selection “multiple” avec parenthèses

```
SELECT * FROM etud
WHERE  (licence=2   OR  licence=1)
     AND  (note < 10   OR  note IS NULL);
SELECT * FROM etud
WHERE  licence=2    OR  (licence=1
     AND  note < 10)   OR   note IS NULL;
```

2 enregistrements

nom	prenom	dateN	note	dep	licence
Martin	Véra	85-10-31	13.5	77	2
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	15.0	77	2
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2
Lefèvre	Laurent		11.5		3

Selection “multiple” avec parenthèses

```
SELECT * FROM etud
WHERE  (licence=2   OR  licence=1)
     AND  (note < 10   OR  note IS NULL);
SELECT * FROM etud
WHERE  licence=2    OR  (licence=1
     AND  note < 10)   OR   note IS NULL;
```

4 enregistrements

nom	prenom	dateN	note	dep	licence
Martin	Véra	85-10-31	13.5	77	2
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	15.0	77	2
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2
Lefèvre	Laurent		11.5		3

Sélection et valeurs nulles

Les lignes de valeurs nulles ne sont pas comprises !

Exemple :

SELECT * FROM etud WHERE note >= 10 OR note < 10;

nom	prenom	dateN	note	dep	licence
Martin	Véra	85-10-31	13.5	77	2
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	15.0	77	2
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2
Lefèvre	Laurent		11.5		3

Pour tout avoir

SELECT * FROM etud WHERE Note>=10 OR Note<10 OR Note = NULL

Table de vérité à 3 valeurs

E	NOT
V	F
F	V
N	N

E1	E2	AND
V	V	V
V	F	F
F	F	F
V	N	N
F	N	F
N	N	N

E1	E2	OR
V	V	V
V	F	V
F	F	F
V	N	V
F	N	N
N	N	N

E	Calcul sur E : + - * /
N	N

Sélection avec comparaison de colonne

Quels étudiants ont des résultats identiques en maths et en bio ?
SELECT * FROM etud1 WHERE note_m = note_b;

nom	prenom	dateN	**note_m**	**note_b**
Martin	Véra	85-10-31	**13.5**	**15.0**
Martin	Annie	85-12-31	**15.5**	**10.0**
Dupont	Sylvie	83-02-03	**15.0**	**15.0**
Martin	Annie	83-10-22		
Dupond	Laurent			**02.0**
Lefèvre	Laurent		**11.5**	

Sélection avec opération

Quels étudiants ont des résultats meilleurs en maths (+ 2 pts) qu'en bio ?
SELECT * FROM etud1 WHERE note_m >= (note_b + 2) ;

nom	prenom	dateN	**note_m**	**note_b**
Martin	Véra	85-10-31	**13.5**	**15.0**
Martin	Annie	85-12-31	**15.5**	**10.0**
Dupont	Sylvie	83-02-03	**15.0**	**15.0**
Martin	Annie	83-10-22		
Dupond	Laurent			**02.0**
Lefèvre	Laurent		**11.5**	

Sélection avec IN

```
SELECT * FROM etud WHERE dep IN (77,92,93);
```

équivalent à :

```
SELECT * FROM etud
    WHERE dep = 77 OR dep = 92 OR dep = 93;
```

nom	prenom	dateN	note	dep	licence
Martin	Véra	85-10-31	13.5	77	2
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	15.0	77	2
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2
Lefèvre	Laurent		11.5		3

Sélection avec NOT IN

SELECT * FROM etud WHERE dep NOT IN (77,92,93);
équivalent à :
SELECT * FROM etud
    WHERE dep <> 77 OR dep <> 92 OR dep <> 93;

nom	prenom	dateN	note	dep	licence
Martin	Véra	85-10-31	13.5	77	2
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	15.0	77	2
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2
Lefèvre	Laurent		11.5		3

ATTENTION : Ligne avec NULL non comptée

Sélection avec intervale

```
SELECT * FROM etud
    WHERE note BETWEEN 11.5 AND 15 ;
```

équivalent à :

```
SELECT * FROM etud
    WHERE note >= 11.5 AND note <=15 ;
```

nom	prenom	dateN	note	dep	licence
Martin	Véra	85-10-31	**13.5**	77	2
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	**15.0**	77	2
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2
Lefèvre	Laurent		**11.5**		3

Sélection avec intervale

```
SELECT * FROM etud
    WHERE note BETWEEN 'D' AND 'M';
```

nom	prenom	dateN	note	dep	licence
Martin	Véra	85-10-31	**13.5**	77	2
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	**15.0**	77	2
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2
Lefèvre	Laurent		**11.5**		3

Sélection avec intevale

```
SELECT * FROM etud
    WHERE date_n BETWEEN '83-02-01' AND '83-02-15'  ;
```

nom	prenom	dateN	note	dep	licence
Martin	Véra	85-10-31	**13.5**	77	2
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	**15.0**	77	2
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2
Lefèvre	Laurent		**11.5**		3

Sélection avec chaîne de caractères

Grâce à LIKE on peut faire des comparaisons avec des caractères jokers :
SELECT * FROM etud WHERE nom LIKE 'Dupon_' ;

nom	prenom	dateN	note	dep	licence
Martin	Véra	85-10-31	13.5	77	2
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	15.0	77	2
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2
Lefèvre	Laurent		11.5		3
Dupontel	Anne				
Mattel	Laurence				

Sélection avec chaîne de caractères

Grâce à LIKE on peut faire des comparaisons avec des caractères jokers :
SELECT * FROM etud WHERE nom LIKE 'Dupon%' ;

nom	prenom	dateN	note	dep	licence
Martin	Véra	85-10-31	13.5	77	2
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	15.0	77	2
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2
Lefèvre	Laurent		11.5		3
Dupontel	Anne				
Mattel	Laurence				

Autres exemples de filtre avec LIKE

- …WHERE prenom LIKE 'Ann_' ;
  - Retrouve : Anne, Anna mais pas Annie, Annette
- … WHERE prenom LIKE 'Laure _ _ _' ;
  - Retrouve : Laurence, Laurette mais pas Laurent, Laure
- … WHERE prenom LIKE 'Ann%' ;
  - Ann, Anne, Annie, Annette…
- … WHERE prenom LIKE '%e' ;
  - Laurence, Sylvie, Anne, Annie,…

Caractères jokers et valeur NULL

- SELECT * FROM etud WHERE dep LIKE '%' ;
  - Les NULL ne sont pas sélectionnés
  - Les valeurs vides sont affichées
- SELECT * FROM etud WHERE dep NOT LIKE '%' ;
  - Les NULL ne sont pas sélectionnés
  - Les valeurs vides ne sont pas sélectionnés
- SELECT * FROM etud WHERE dep IS NULL ;
  - Les NULL sont sélectionnés
  - Les valeurs vides ne sont pas sélectionnées

Table de vérité du NULL
NOT NULL -> NULL

SELECT et opérations

```
SELECT nom, prenom, CONCAT(nom,'  ', prenom)
    FROM etud  ;
```

nom	prenom	CONCAT(nom,' ',prenom)
Martin	Véra	Martin Véra
Martin	Annie	Martin Annie
Dupont	Sylvie	Dupont Sylvie
Martin	Annie	Martin Annie
Dupond	Laurent	Dupond Laurent
Lefèvre	Laurent	Lefèvre Laurent

SELECT et alias

```
SELECT nom, prenom,
    CONCAT(nom,'  ', prenom) AS 'Nom complet'
    FROM etud;
```

nom	prenom	Nom complet
Martin	Véra	Martin Véra
Martin	Annie	Martin Annie
Dupont	Sylvie	Dupont Sylvie
Martin	Annie	Martin Annie
Dupond	Laurent	Dupond Laurent
Lefèvre	Laurent	Lefèvre Laurent

SELECT et opérations

SELECT upper(nom), prenom, note, note*2 FROM etud;

upper(nom)	prenom	note	note*2
MARTIN	Véra	13.5	27.0
MARTIN	Annie	15.5	31.0
DUPONT	Sylvie	15.0	30.0
MARTIN	Annie	05.7	11.4
DUPOND	Laurent		
LEFÈVRE	Laurent	11.5	23.0

En majuscule

Note sur 40

Autres exemples de champs calculable :
SELECT salaire + prime …
SELECT prix * 19.6 / 100
SELECT prix - remise

Fonctions de groupe

- Permet de considerer les réponses dans leur ensemble
- Fonctions de regroupement : on brasse l'ensemble, le détail est perdu
- Mais on peut obtenir des totaux, des moyennes, diverses statistiques sur cet ensemble
- Par exemple, quelle est la meilleur note ? Combien y'a t'il de clients ?
- Mais on ne peut pas trouver directement « Qui a eu la meilleure note »

Fonctions de groupe

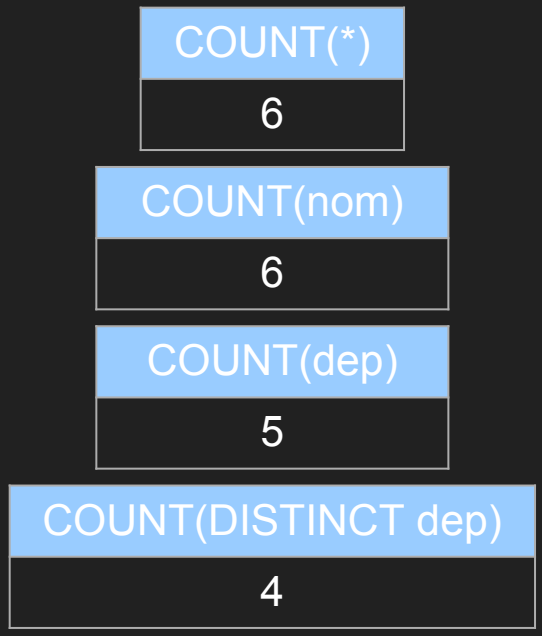
- COUNT : nombre d'enregistrements
  - Ex : nombre d'étudiants total
  - Ex : nombre d'étudiants dont la note est connue
- SUM : somme
  - Ex : somme des prix dans une facture
- MAX :maximum
  - Ex : liste des étudiants qui ont la meilleure note

Fonctions de groupe

- MIN : minimum
  - Ex : liste des étudiants qui ont la note la plus basse
- AVG : moyenne (average) pour des nombres
  - Ex : moyenne des notes de licence 1

Fonction de groupe : détail

- SELECT COUNT(*) FROM etud ;
  - Compte les enregistrements (même Null)
- SELECT COUNT(nom) FROM etud ;
  - Nb d'enregistrements de nom "non Null"
- SELECT COUNT(dep) FROM etudiant ;
  - Nb d'enregistrements de département "non Null"
- SELECT COUNT(DISTINCT dep) FROM etud ;
  - Les doublons ne sont comptés qu'une fois



Fonction de groupe et alias

- SELECT COUNT(*)
  FROM etud ;

COUNT(*)
6

- SELECT COUNT(*) AS 'Nb d\'étudiants' FROM etud ;

Nb d'étudiants
6

- SELECT dep AS Département,
  nom AS Nom,
  prenom AS Prénom FROM etud;

Département	Nom	Prénom
...	...	....

- AS est facultatif (selon les implémentations)

```
SELECT dep Département,
    nom Nom, prenom Prénom
    FROM etud;
```

Exemple sur les fonctions de groupe

- SELECT AVG(note) FROM etud ;
  - Moyenne des notes (non NULL)

AVG(note)
12.24

- SELECT MIN(note), AVG(note), MAX(note) FROM etud;

MIN(note)	AVG(note)	MAX(note)
5.7	12.24	15.5

Erreurs sur les fonctions de groupe

- SELECT nom, MIN(note) FROM etud ;
  - nom, note des étudiants ayant la note la plus basse
- Incorrect :
  - pas de mélange entre Agrégat et Champ !

Regroupement de données : GROUP BY

Pour la lisibilité de la réponse

```
SELECT dep, count(*) FROM etud
    GROUP BY dep ;
```

nom	prenom	date_n	note	dep	licence
Martin	Véra	85-10-31	13.5	77	2
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	15.0	77	2
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2
Lefèvre	Laurent		11.5		3

dep	count(*)
null	1
75	1
77	2
92	1
93	1

Erreur d'utilisation sur GROUP BY

SELECT nom, prenom FROM etud GROUP BY dep;
SELECT * FROM etud GROUP BY dep;

- Même problème que les fonctions d'aggrégation

nom	prenom	date_n	note	dep	licence
Martin	Véra	85-10-31	13.5	77	2
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	15.0	77	2
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2
Lefèvre	Laurent		11.5		3

Condition sur le regroupement : HAVING

- Ressemble au WHERE mais appliqué au regroupement

Départements où résident plus de 20 étudiants

```
SELECT dep, count(num)
    FROM etud
    GROUP BY dep
    HAVING count(num) > 20
```

dep	count(num)
75	500
76	10
77	612

- S'il y a une clause WHERE, elle sera appliquée avant le calcul d'agrégat.

Récapitulation : exercice

Par année de licence, présenter par moyenne croissante, le nombre d'étudiants inscrits, la moyenne, le min, le max.

nom	prenom	dateN	note	dep	licence
Martin	Véra	85-10-31	**13.5**	77	2
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	**15.0**	77	2
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2
Lefèvre	Laurent		**11.5**		3

Récapitulation : exercice

```
SELECT licence,
        COUNT(*)    AS Nb,
        AVG(note)   AS Moyenne,
        MIN(note)   AS Min,
        MAX(note)   AS Max
FROM etud GROUP BY licence ORDER BY Moyenne ;
```

licence	Nb	Moyenne	Min	Max
1	2	10.60000	05.7	15.5
3	1	11.50000	11.5	11.5
2	2	14.25000	13.5	15.0

Récapitulation : exercice 2

Par année de licence, présenter par moyenne croissante, le nombre d'étudiants notés, la moyenne, le min, le max.

nom	prenom	dateN	note	dep	licence
Martin	Véra	85-10-31	**13.5**	77	2
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	**15.0**	77	2
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2
Lefèvre	Laurent		**11.5**		3

Récapitulation : exercice

```
SELECT licence,
        COUNT(note)    AS Nb,
        AVG(note)      AS Moyenne,
        MIN(note)      AS Min,
        MAX(note)      AS Max
FROM etud GROUP BY licence ORDER BY Moyenne ;
```

licence	Nb	Moyenne	Min	Max
1	2	10.60000	05.7	15.5
3	1	11.50000	11.5	11.5
2	2	14.25000	13.5	15.0

Copie de table

- Syntaxe générale
  - CREATE TABLE table2 AS *requête* ;
- Exemples :
  - Copie de la table des étudiants de licence
    - CREATE TABLE etud_copie AS SELECT * FROM etud ;
  - Copie partielle de table : uniquement les étudiants de licence 1
    - CREATE TABLE etud_lic1

      AS SELECT nom, prenom, date_n, note

      FROM etud

      WHERE licence=1 ;

Les jointures

Exemple de BD à tables multiples

etud

nom	prenom	date_n	note	dep	licence
Martin	Véra	85-10-31	13.5	77	2
Martin	Annie	85-12-31	15.5	75	1
Dupont	Sylvie	83-02-03	15.0	77	2
Martin	Annie	83-10-22	05.7	93	1
Dupond	Laurent			92	2
Lefèvre	Laurent		11.5		3

departement

dep_num	dep_nom
01	Ain
…	...
75	Paris
77	Seine-et-Marne
92	Hauts-de-Seine
93	Seine-Saint-Denis

Produit cartésien de table

SELECT * FROM etud, departement

nb enregistrements =
6 étudiants x
n départements

nom	prenom	dateN	note	dep	licence	dep_num	dep_nom
Martin	Véra	85-10-31	13.5	77	2	01	Ain
Martin	Annie	85-12-31	15.5	75	1	01	Ain
Dupont	Sylvie	83-02-03	15.0	77	2	01	Ain
Martin	Annie	83-10-22	05.7	93	1	01	Ain
Dupond	Laurent			92	2	01	Ain
Lefèvre	Laurent		11.5		3	01	Ain
....	....	...	...	..	...	...	...

Jointure de table avec SELECT

```
SELECT * FROM etud, departement
      WHERE dep=dep_num ;
SELECT * FROM etud, departement
      WHERE etud.dep=departement.dep_num ;
SELECT * FROM etud e, departement d
      WHERE e.dep=d.dep_num ;
```

Ne pas oublier la clause WHERE

Attention aux NULL

etud

departement

nom	prenom	dateN	note	dep	licence	dep_num	dep_nom
Martin	Véra	85-10-31	13.5	77	2	77	Seine-et-Marne
Martin	Annie	85-12-31	15.5	75	1	75	Paris
Dupont	Sylvie	83-02-03	15.0	77	2	77	Seine-et-Marne
Martin	Annie	83-10-22	05.7	93	1	93	Seine-Saint-Denis

Notation pointée des champs

Les champs peuvent être préfixés pour le nom de la table ou de son alias
SELECT etud.nom FROM etud WHERE etud.note>10 ;
SELECT e.nom FROM etud e WHERE e.note>10 ;

Notation obligatoire quand les noms sont identiques dans les champ de jointure
SELECT * FROM etud, departement
WHERE etud.dep=departement.dep ;

dans les champs à afficher
SELECT nom, etud.dep FROM etud, departement
WHERE etud.dep=departement.dep ;

dans une auto-jointure
SELECT * FROM etud e1, etud e2
WHERE e1.num=e2.binome ;

Jointure avec JOIN (SQL2)

Produit cartésien

```
SELECT * FROM etud, departement ;
SELECT * FROM etud CROSS JOIN departement ;
SELECT * FROM etud JOIN departement ;
```

Jointure

```
SELECT *
      FROM etud, departement
      WHERE dep=dep_num ;
SELECT *
      FROM etud
      JOIN departement
            ON dep=dep_num ;
```

```
SELECT * FROM t1, t2, t3
      WHERE t1.a=t2.a AND t2.b=t3.b;

SELECT * FROM t1
      JOIN t2
            ON t1.a=t2.a
      JOIN t3
            ON t2.b=t3.b;
```

Jointure avec sélection

```
SELECT * FROM etud, departement
    WHERE dep=dep_num
    AND  (dep=77 or dep=75) AND licence=2 ;
```

Ne pas oublier la clause WHERE de jointure

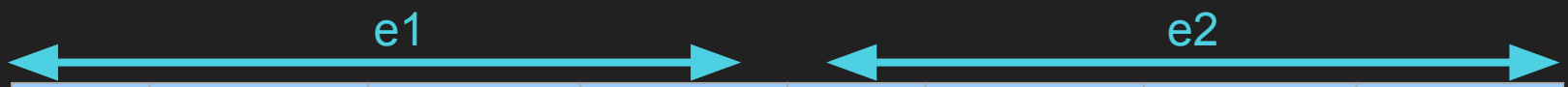
etud ⟷ departement

nom	prenom	dateN	note	dep	licence	dep_num	dep_nom
Martin	Véra	85-10-31	13.5	77	2	77	Seine-et-Marne
Martin	Annie	85-12-31	15.5	75	1	75	Paris
Dupont	Sylvie	83-02-03	15.0	77	2	77	Seine-et-Marne
Martin	Annie	83-10-22	05.7	93	1	93	Seine-Saint-Denis
Dupond	Laurent			92	2	92	Hauts-de-Seine

Auto jointure de table

Jointure de la table sur elle-même
Utilisation de 2 alias pour la même table

```
SELECT * FROM etud e1, etud e2
    WHERE e1.num=e2.binome ;
```



num	nom	prenom	binome	num	nom	prenom	binome
1	Martin	Véra	2	2	Martin	Annie	1
2	Martin	Annie	1	1	Martin	Véra	2
3	Dupont	Sylvie	4	4	Martin	Annie	3
4	Martin	Annie	3	3	Dupont	Sylvie	4
5	Dupond	Laurent	6	6	Lefèvre	Laurent	5

Jointure naturelle

Jointure avec les colonnes communes de même nom
SELECT * FROM etud NATURAL JOIN departement
SELECT * FROM etud e, departement d
    WHERE e.dep=d.dep ;

etud

departement

nom	prenom	dateN	note	dep	licence	dep_num	dep_nom
Martin	Véra	85-10-31	13.5	77	2	77	Seine-et-Marne
Martin	Annie	85-12-31	15.5	75	1	75	Paris
Dupont	Sylvie	83-02-03	15.0	77	2	77	Seine-et-Marne
Martin	Annie	83-10-22	05.7	93	1	93	Seine-Saint-Denis
Dupond	Laurent			92	2	92	Hauts-de-Seine

Jointure avec restriction

```
SELECT * FROM etud, departement
    WHERE dep=dep_num
        AND note > 10 ;
```

```
SELECT * FROM etud JOIN departement
ON dep=dep_num
WHERE note > 10 ;
```

Méthodologie d’écriture de SELECT

- Liste ordonnée des champs à afficher
  - SELECT champ1,champ2, fonction de groupe
  - Préfixer les champs dont le nom est identique dans 2 tables
- Liste des tables dont on cherche les champs
  - Jointure nécessaire ?
    - FROM table1,table2 WHERE condition de jointure
    - FROM table1 JOIN table2 ON condition de jointure
- Conditions de recherche sur les enregistrements ?
  - WHERE condition

Méthodologie d’écriture de SELECT

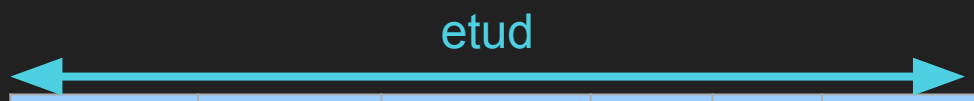
- Fonction de groupe ?
  - GROUP BY …, …
- Conditions de recherche sur les groupes
  - HAVING condition
- Ordre de présentation des enregistrements ?
  - ORDER BY champ1, alias2, (alias si fonction d'agrégat)
- Ajouter des alias si nécessaire (présentation, agrégat, autojointure…)

Jointure “externe”

- Les jointures vues précédemment sont des jointures dites interne
- Les jointures externes rajoutent les lignes d’ une des 2 tables non sélectionnées
- 3 jointures externes
  - Droite
  - Gauche
  - Totale

Jointure externe gauche

SELECT * FROM etud LEFT OUTER JOIN departement
ON dep=dep_num ;



nom	prenom	dateN	note	dep	licence	dep_num	dep_nom
Martin	Véra	85-10-31	13.5	77	2	77	Seine-et-Marne
Martin	Annie	85-12-31	15.5	75	1	75	Paris
Dupont	Sylvie	83-02-03	15.0	77	2	77	Seine-et-Marne
Martin	Annie	83-10-22	05.7	93	1	93	Seine-Saint-Denis
Dupond	Laurent	null	null	92	2	92	Hauts-de-Seine
Lefèvre	Laurent	null	11.5	null	3	null	null

Jointure externe droite

```
SELECT * FROM etud RIGHT OUTER JOIN departement
    ON dep=dep_num ;
```

etud						departement	
nom	prenom	dateN	note	dep	licence	dep_num	dep_nom
Martin	Véra	85-10-31	13.5	77	2	77	Seine-et-Marne
Martin	Annie	85-12-31	15.5	75	1	75	Paris
Dupont	Sylvie	83-02-03	15.0	77	2	77	Seine-et-Marne
Martin	Annie	83-10-22	05.7	93	1	93	Seine-Saint-Denis
Dupond	Laurent	null	null	92	2	92	Hauts-de-Seine
null	null	null	null	null	null	07	Ain
null	null	null	null	null	null	07	Ardèche
Lefèvre	Laurent	null	11.5	**null**	3	**null**	null

Jointure externe et fonction d'aggrégat

```
SELECT dep_num, count(nom)
FROM etud
RIGHT OUTER JOIN departement
ON dep=dep_num
GROUP BY dep_num ;
```

dep	count(*)
01	0
07	0
75	1
77	2
92	1
93	1

```
SELECT dep_num, count(nom)
FROM etud, departement
WHERE dep=dep_num
GROUP BY dep_num;
```

dep	count(*)
01	0
07	0
75	1
77	2
92	1
93	1

Equi-jointure & théta-jointure

- Equi-jointure : =
- Théta-jointure : > >= < <=

- SELECT * FROM salarie s
  JOIN responsable r
  ON s.salaire >= r.salaire

Aller un peu plus loin

Requêtes imbriquées

SELECT * FROM …
    WHERE <condition portant sur une autre requête>

SELECT dep
    FROM departement
    WHERE dep NOT IN
        (SELECT DISTINCT dep_num FROM etud);

Permet de résoudre des questions courantes :
SELECT nom
    FROM etud
    WHERE note = (SELECT MAX(note) FROM etud)

Exercices :
    Qui habite dans le même département que Sylvie Dupont ?
    Combien d'étudiant ont eu la plus mauvaise note ?

Opérations ensemblistes

Les tables sont considérées comme des ensembles
On peut donc faire des opérations GLOBALES sur ces tables
Les operations portent sur les TUPLES, on doit donc manipuler des ensembles structurés à l'identique

Les opérations sont les suivantes :

- UNION : regroupe les résultats de 2 requêtes (les résultats doivent être identiques)
- INTERSECT : conserve les lignes identiques des 2 requêtes
- MINUS
- EXISTS , EXCEPT ou NOT EXISTS

Opérations ensemblistes

- Un exemple :
  - SELECT nom FROM etud WHERE dep=77

    UNION

    SELECT nom FROM ancien_etud WHERE dep=77 ;
  - SELECT nom,prenom,numINSEE FROM etud

    INTERSECT

    SELECT nom,prenom,numINSEE FROM ancien_etud;
- Que font ces deux réquêtes ? (ancien_etud: table des anciens étudiants, identique à étud)

Opérations ensemblistes : EXISTS

EXISTS va renvoyer VRAI si la requête qu'on lui soumet retourne au moins un tuple...
SELECT ……. WHERE EXISTS (SELECT ….)
SELECT …. WHERE NOT EXISTS (SELECT ….)

La sous requête est indépendante, donc il faut trouver un moyen de faire un lien avec la requête principale.

Exemple
SELECT * FROM dep
WHERE NOT EXISTS (select * FROM etud
WHERE dep_num=dep)

Que fait cette requête ?
Que remarquez vous dans la sous requete ?
Quels particularités par rapport à un NOT IN ?

Fonction de codage MySQL

- PASSWORD()
- ENCRYPT()
- ENCODE()
- DECODE ()
- MD5()

Retour sur les contraintes des données

- Présence de valeur : NOT NULL
- Contrainte de valeur : PRIMARY KEY
- Unicité de valeur : UNIQUE
- Contrainte de valeur : CHECK BETWEEN 0 AND 20
- Intégrité référentielle : FOREIGN KEY … REFERENCES …

Avec les foreign key

- Triggers :
  - ON DELETE
  - ON UPDATE
- Action sur les triggers :
  - CASCADE
  - SET DEFAULT
  - SET NULL
  - RESTRICT

Appliquer les contraintes

Méthode 1 : lors de la création de la table

```
CREATE TABLE etudiant (
        num INT PRIMARY KEY,
        login VARCHAR(15) UNIQUE,
        nom VARCHAR(15) NOT NULL,
        codeDep CHAR(3),
        note INT CHECK (note BETWEEN 0 AND 20),
        FOREIGN KEY codeDep REFERENCES departement(numDep)
        ON UPDATE CASCADE
 );
```

Méthode 2 : idem

```
CREATE TABLE etudiant (
        num INT,
        …,
        PRIMARY KEY (num);
```

Méthode 3 : Après, on peut ajouter la contrainte en la nommant

```
CONSTRAINT chk_Note CHECK (note between 0 AND 20)
```

Bibliographie - Sitographie

- Ouvrages
  - SQL2 - Initiation/programmation.-
  - C. Marée & G. Ledant.- InterEditions, 1999
  - SQL Développement.- F. Brouard.- CampusPress, 2001.- Col. Référence Développement
- Sites
  - http://sqlpro.developpez.com (Frédéric Brouard)