Le langage SQL

16 mars 2011

Florence Petit/ Sylvain Cherrier





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)
- 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 - Evolution

- 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

Classification des commandes du langage SQL

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

SQL

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





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  …. ;
```





Exemple de table : 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





NULL : absence de valeur

- Valeur inconnue

  ex : note, date de naissance…
- 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

Ecriture du code 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 ;
```

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
- 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
- Des commentaires sont possibles sous plusieurs formes :

  /* commentaire sur plusieurs lignes */

  -- espace puis commentaire sur une ligne

  # commentaire sur une ligne





Création de table avec CREATE

- Exemple

```
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)
    ) ;
```





Suppression de table avec DROP

- Syntaxe générale

  DROP TABLE *nom_table* ;





Suppression d'enregistrements : DELETE

- 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' ;





Insertion d'enregistrements : INSERT

- Syntaxe générale n°1

```
INSERT INTO nom_table
    VALUES (val1, val2,…,val7) ;
```

- Exemple de syntaxe n°1

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

- Syntaxe générale n°2

```
INSERT INTO nom_table (champ3, champ2,champ7)
    VALUES (val3, val2, val7) ;
```

- Exemple de syntaxe n°2

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





Mise à jour : UPDATE… SET

- 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 base de données : 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

Quelques exemples simples de SELECT

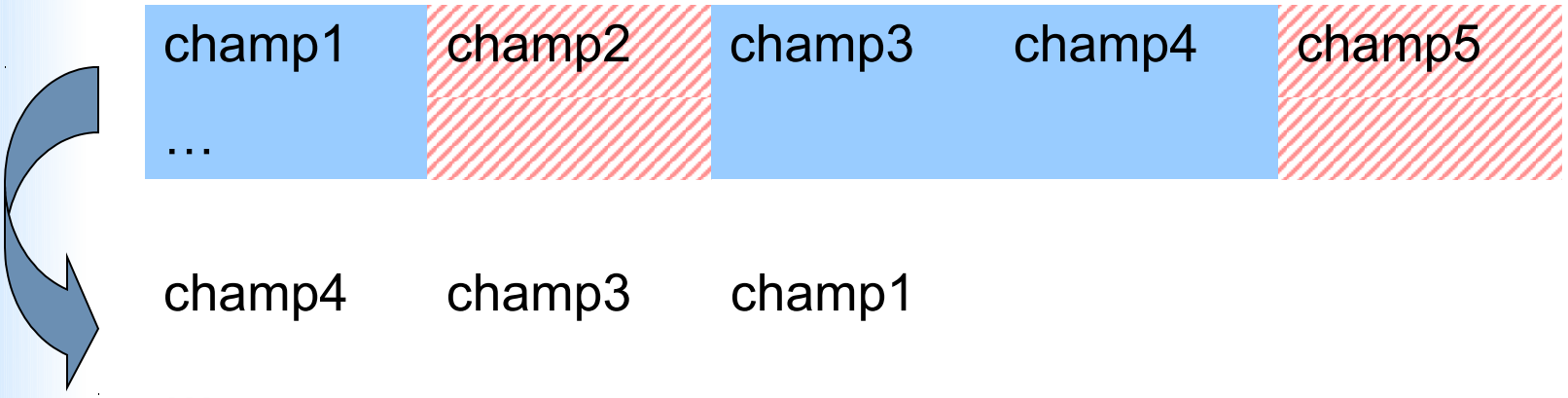
- SELECT * FROM table1;

  On obtient toute la table

- SELECT champ3 FROM table1;

  On obtient uniquement le champ3 de la table

- SELECT champ4, champ3, champ1 FROM table1;



Syntaxe du SELECT

- Syntaxe minimale

  SELECT … FROM table;

- Syntaxe complète (respecter l'ordre des clauses)

```
SELECT
    [DISTINCT | ALL  ] {* | champ [[AS] alias], ... }
    FROM table [[AS] alias], ...
    [WHERE { condition | sous condition} ]
    [GROUP BY champ , ...]
    [HAVING condition]
    [ORDER BY {champ | num}{ASC | DESC}, ...]
    [LIMIT [deb,] nb];
```

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 de toutes les colonnes de la table

  SELECT * FROM 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 de noms de colonnes

Prénom	Nom de famille
...	...

- 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;
```





Caractère d'échappement

- SELECT nom 'Nom d'étudiant' ... -> erreur

- SELECT nom "Nom d'étudiant" ...
- SELECT nom 'Nom d''étudiant' ...
- SELECT nom 'Nom d\'étudiant' ...

Gérer les doublons : ALL (facultatif) et 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

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

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 sur colonnes multiples

- 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

Tri des lignes : ORDER BY

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 d'enregistrements affichés

- 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

Prélever n ligne(s) à partir de la ligne i + 1

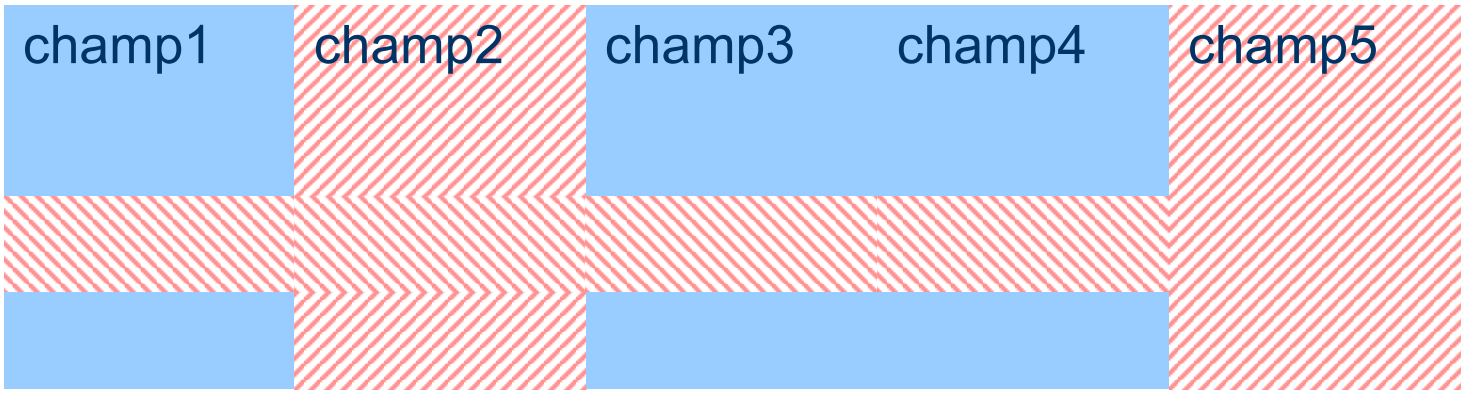
- SELECT * FROM etud LIMIT i,n ;
- SELECT * FROM etud ORDER BY note DESC LIMIT 3,2 ;
- Les lignes sont notées à partir de i=0 et non de i=1 !

i	rang	nom	prenom	dateN	note	dep	licence
0	1	Martin	Annie	85-12-31	15.5	75	1
1	2	Dupont	Sylvie	83-02-03	15.0	77	2
2	3	Martin	Véra	85-10-31	13.5	77	2
3	4	Lefèvre	Laurent		11.5		3
4	5	Martin	Annie	83-10-22	05.7	93	1
5	6	Dupond	Laurent			92	2

2 enregistrements

SELECT …. FROM ….WHERE ….

- SELECT -> sélection de colonnes (projection)
- WHERE -> sélection de lignes (sélection = restriction)
- Exemples :
  - SELECT * FROM t1 WHERE nom='Martin';
  - SELECT * FROM t1 WHERE note >=10;
  - SELECT * FROM t1 WHERE note >10 AND dep=77;



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

Expression des valeurs constantes

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

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 AND

- 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 OR

- 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

Ajout de parenthèses si besoin

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

B
```
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

A

B

Sélection et valeur Null

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

  -> Les lignes de valeurs NULL ne sont pas comptées !

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

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

  -> pour retrouver toutes les valeurs, y compris NULL

Tables de vérité à 3 valeurs (dont NULL)

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

Opérations algébriques avec NULL

E	Calcul sur E : + - * /
N	N





Sélection avec comparaison de colonnes

- 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 sur les colonnes

- 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**	

Liste de valeurs avec IN

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

Pour les chaînes de caractères, ('val1','val2','val3')

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

Liste de valeurs avec NOT IN

- SELECT * FROM etud
  WHERE dep NOT IN (77,92,93);

équivalent à :

- SELECT * FROM etud
  WHERE dep<> 77 AND dep<>92 AND 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

Null
non compté

Intervalle avec BETWEEN (nombre)

- 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

- Respecter l'ordre des bornes (inférieure puis supérieure)

Intervalle avec BETWEEN (chaîne)

- SELECT * FROM etud
  WHERE nom 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

Intervalle avec BETWEEN (date)

- SELECT * FROM etud
  WHERE date_n BETWEEN '83-02-01' AND '83-02-15' ;
- Ne pas oublier les '...'

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





Caractère _ pour un caractère et un seul

- SELECT * FROM etud
  WHERE nom LIKE 'Dupon_' ;

Attention LIKE au lieu de =

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				

Caractère % pour 0, 1 ou n caractère(s)

- SELECT * FROM etud
  WHERE nom LIKE 'Dupon%' ;

Attention LIKE au lieu de =

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				





Exemples de filtres avec LIKE

- …WHERE prenom LIKE 'Ann_' ;
- -> Anne, Anna mais pas Annie, Annette

- … WHERE prenom LIKE 'Laure _ _ _' ;
- -> Laurence, Laurette mais pas Laurent, Laure

- … WHERE prenom LIKE 'Ann%' ;
- -> Ann, Anne, Annie, Annette…

- … WHERE prenom LIKE '%e' ;
- -> Laurence, Sylvie, Anne, Annie,…





Caractères Joker et valeur Null

Table de vérité du NULL

NOT NULL -> 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 sélectionnés

Opérateurs

Opérateurs arithmétiques	+ - * / % (MOD)
Opérateurs de comparaison	
égalité / non égalité	= <> !=
supériorité / infériorité	> < >= <=
appartenance à une liste de valeurs	IN ('val1','val2','val3')
intervalle de valeurs	BETWEEN val1 AND val2
motif de chaîne	LIKE % _
valeur Null	IS NULL IS NOT NULL
Opérateurs logiques	
et non ou ou exclusif	&& ! || XOR
Connecteurs logiques	AND OR NOT
Gestion de la priorité	( )
Caractère d'échappement (avant ' " %)	\

Ordre de priorité des opérateurs

NOT

- (négatif)

^

* / % MOD

+ -

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

BETWEEN

AND &&

OR ||





Concaténation : CONCAT(exp1, exp2, exp3)

- 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





Fonction de concaténation et alias

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

Exemples avec fonction et calcul

- 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

mise en majuscule

pour obtenir une note sur 40

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





Fonctions de groupe (d'agrégat ou d'ensemble)

- COUNT nombre d'enregistrements
  - nombre d'étudiants total
  - nombre d'étudiants dont la note est connue
- SUM somme
  - somme des prix dans une facture
- MAX maximum
  - liste des étudiants qui ont la meilleure note
- MIN minimum
  - liste des étudiants qui ont la note la plus basse
- AVG moyenne (average) pour des nombres
  - moyenne des notes de licence 1





Que comptent les fonctions d'agrégation ?

- 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

COUNT(*)
6

COUNT(nom)
6

COUNT(dep)
5

COUNT(DISTINCT dep)
4

Alias pour les noms de champs : AS

- SELECT COUNT(*)
  FROM etud ;

COUNT(*)
6

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

Nb d'étudiants
6

  -> s'il y a des espaces, entourer de '…' ou "…"

- 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;





Erreur d'utilisation des fonctions d'agrégation

- 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

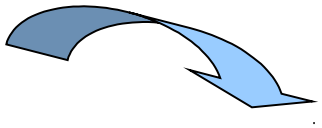
Incorrect :

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

Groupements 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

Groupements de données : exercices

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

Erreur d'utilisation du GROUP BY

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

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.





Méthodologie d'écriture de requête 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
- 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…)

Quelques fonctions

- ROUND(nb,n) -> arrondi à n décimales
  - ROUND(24.16, 1) = 24.2
  - ROUND(24.16, 0) = ROUND(24.16) = 24
- RAND(nb) -> nb aléatoire de 0 à 1 (plafonné à 0,n)
- UPPER(col) UCASE(col) -> chaîne en majuscule
- LOWER(col) LCASE(col) -> chaîne en minuscule
- CONCAT(exp1,exp2…) -> concaténation
- LENGTH(col) -> Nb de caractères
- SUBSTRING(col,n,p) -> sous-chaîne (du $n^{ème}$, Lg=p)





Fonctions de contrôle (test)

- IF(condition,val_vrai,val_faux)
  ex : IF(note>=10,'admis','refusé')
- IFNULL(valeur,val_faux)
  ex : IFNULL(note,'abs')
- CASE val WHEN comp1 THEN res1
  WHEN comp2 THEN res2
  ELSE res3
  END

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 (n^{ème} jour du mois)
- DAYOFYEAR() -> 310 (n^{ème} 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

Motifs de format pour DATE_FORMAT(date,'format')

- %y an (2 chiffres) : 97
- %m mois (01 à 12) : 04
  %b mois (court) : Apr
- %d jour (01 à 31) : 08
- %a jour (court) : Sat
  %w jour de semaine (Di=0, Lu=6)
- %j N° jour (1 à 365)
- %U semaine de l'année (début=di)
- %r heure (12h) : 11:59:30 PM
- %I heure (12h) : 11
- %p AM ou PM
- %i minutes (2 chiffres) : 51
- %S secondes (2 chiffres) : 53

%Y an (4 chiffres) : 1997

%M mois (long) : April

%e jour (1 à 31) : 8

%W jour (long) : Saturday

%u idem (début=lu)

%T heure (24h) : 23:59:30

%H heure (24h) : 23





Copie totale/partielle de table avec CREATE

- Syntaxe générale

```
CREATE TABLE table2
    AS requête ;
```

- 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 ;
```

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 ;

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

Jointure de table avec WHERE (SQL1)

- 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
Dupond	Laurent			92	2	92	Hauts-de-Seine
Lefèvre	Laurent	null	11.5	null	3	null	null
null	null	null	null	null	null	01	Ain

Produit cartésien de table

nb enregistrements =
6 étudiants x
n départements

- SELECT * FROM etud, departement ;

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
....	....	...	...	..	...	...	...
Martin	Véra	85-10-31	13.5	77	2	77	Seine-et-Marne
Martin	Annie	85-12-31	15.5	75	1	77	Seine-et-Marne
Dupont	Sylvie	83-02-03	15.0	77	2	77	Seine-et-Marne
Martin	Annie	83-10-22	05.7	93	1	77	Seine-et-Marne
Dupond	Laurent			92	2	77	Seine-et-Marne
Lefèvre	Laurent		11.5		3	77	Seine-et-Marne
....	....	...	...	..	...	...	...

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 restriction
  - SELECT * FROM etud, departement
    WHERE dep=dep_num
    AND note > 10 ;
  - SELECT * FROM etud JOIN departement
    ON dep=dep_num
    WHERE note > 10 ;

Jointure de table et sélection d'enregistrement

- 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
Lefèvre	Laurent		11.5		3		

Jointure naturelle de table

- 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	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
Lefèvre	Laurent		11.5	null	3	null	

Jointure externe (FULL | LEFT | RIGHT OUTER JOIN …ON…)

- SELECT * FROM etud LEFT 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
Lefèvre	Laurent	null	11.5	null	3	null	null
null	null	null	null	null	null	01	Ardèche

- par défaut, une jointure est une jointure interne (INNER JOIN)

Jointure externe droite (RIGHT OUTER JOIN …ON…)

- 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'agrégat

- 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

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 ;

e1 e2

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
6	Lefèvre	Laurent	5	5	Dupond	Laurent	6

Auto-jointure de table

- Etudiants dont la note est inférieure à la note de l'étudiant n° 4
- SELECT e1.*
  FROM etud e1, etud e2
  WHERE e1.note < e2.note
  AND e2.num=4 ;

e1 e2

num	nom	prenom	Note	num	nom	prenom	binome
1	Martin	Véra	13.5	2	Martin	Annie	15.5
2	Martin	Annie	15.5	1	Martin	Véra	13.5
3	Dupont	Sylvie	15.0	3	Dupont	Sylvie	15.0
4	Martin	Annie	05.7	4	Martin	Annie	05.7
5	Dupond	Laurent	null	5	Dupond	Laurent	null
6	Lefèvre	Laurent	13.5	6	Lefèvre	Laurent	13.5





Equi-jointure et théta-jointure

- Equi-jointure =

- Théta-jointure > >= < <=

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

Requête imbriquée

- 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 ?





Operations 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

  INTERSECT

  MINUS

  EXISTS , EXCEPT ou NOT EXISTS





Ensemblistes : UNION, INTERSECT

- Un exemple :

  SELECT nom FROM etud WHERE dep=77
  UNION
  SELECT nom FROM ancien_etudiant WHERE dep=77 ;

- SELECT nom,prenom,numINSEE FROM etud
  INTERSECT
  SELECT nom,prenom,numINSEE FROM ancien_etudiant ;

- Que font ces deux réquêtes ? (*ancien_etudiant est une table qui contient la liste des étudiants qui étaient présents chez nous, sa structure est identique à étud*)

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 ?





Commande ALTER : modification de structure

- ALTER TABLE matable
  DROP nomchamp1;
- ALTER TABLE matable
  ADD nomchamp1 VARCHAR(10)
- ALTER TABLE matable1
  ADD FOREIGN KEY (nomchamp1) REFERENCES matable2 (nomchamp2);
- ALTER TABLE matable1
  MODIFY nomchamp VARCHAR(20)
- ALTER TABLE matable1
  CHANGE oldnomchamp newnomchamp VARCHAR(20)





non reconnu a la fac

- concatenate() ||
- cast(champ as float)





Fonctions de codage de MySQL

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





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)





Sommaire du cours SQL

- Généralités
  - Présentation du SQL
  - Exemple de table de données
  - Null
- Sélections (sur monotable) -> SELECT
  - sélection de colonnes
  - tri -> ORDER BY
  - sélection de lignes avec opérateurs
  - fonctions de groupe ou d'agrégation
  - alias -> AS
  - Quelques fonctions
- Création de table et copie -> CREATE
- Suppression de table ->DROP
- Suppression d'enregistrement ->DELETE
- Insertion de données -> INSERT
- Mise à jour de données -> UPDATE





Hors cursus BD1

- Contrôle de transaction (COMMIT, ROLLBACK)
- Restriction d'accès, privilèges (GRANT, REVOKE)
- Optimisation des requêtes
- Index
- Vues (VIEW)
- Jointure externe (LEFT OUTER JOIN)
- Requêtes imbriquées, sous-requêtes
- Clauses HAVING, EXISTS, MATCH, CHECK
- Trigger
- Opérateur ensembliste (union, intersection, différence)
- Division
- Modification de schéma (ALTER)
- SQL programmé





Principales commandes du SQL

- Sélection de champs/lignes
```
SELECT col1, col2, col3
FROM table1
WHERE <condition>;
```
- Mise à jour de données
```
UPDATE table1
SET col1=val1,  col2='val2'
WHERE <condition>;
```
- Insertion de lignes
```
INSERT  INTO table1
VALUES (val1, 'val2');
```
- Suppression de lignes
```
DELETE
FROM table1
WHERE <condition>;
```
- Suppression de table
```
DROP  TABLE table1;
```
- Création de table
```
CREATE  TABLE table1 (
col1 type1 PRIMARY KEY,
col2 type2
);
```





Les contraintes

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

Les contraintes

- Valeur obligatoire : NOT NULL
- Unicité de valeur : UNIQUE
- Clé primaire (unicité + not null) : PRIMARY KEY
- Présence de valeur : PRIMARY KEY
- Contrainte de domaine : CHECK (note between 0 AND 20)
- Intégrité référentielle : FOREIGN KEY … REFERENCES …

- Les triggers (déclencheurs, gachettes)
  ON DELETE
  ON UPDATE
- avec leurs actions :
  CASCADE
  SET DEFAULT
  SET NULL
  RESTRICT





Les contraintes : exemple de SQL

- Méthode 1

```
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

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

- Méthode 3 : on peut ajouter la contrainte en la nommant

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