

L1 Sciences Fondamentales, S2 Langage SQL

Jean-François COUCHOT
Université de Franche-Comté, UFR-ST



Plan

Introduction

Data Query Language

Data Manipulation Language

Data Definition Language

Data Control Language



Devoir surveillé en Bases de Données



Date : 06 mars de 11h à 12h30 en Amphi A.

Matériel autorisé : une feuille manuscrite de notes personnelles.



Plan



Introduction

Data Query Language

Data Manipulation Language

Data Definition Language

Data Control Language



Langages de SGBD relationnelle : une histoire

De l'Alpha à ...SQL

- ▶ Alpha, inventé par Edgar F. Codd (1970)
- ▶ ...SEQUEL – Structured English QUery Language ...
- ▶ SQL – Structured Query Language (1974)

Quelques versions de SQL

- ▶ SQL-86 (SQL1) : norme proposée par l'ANSI¹
 - ▶ Remarque : permet de manipuler des données mais pas de modifier un modèle relationnel
- ▶ SQL-92 (SQL2) : version utilisée dans la plupart des SGBD actuels
 - ▶ plus de types de données gérés et permet de modifier un modèle relationnel
- ▶ SQL-99 (SQL3) : version qui intègre un modèle objet-relationnel

1. American National Standard Institute

Structured Query Language



SQL : un langage de requêtes structuré, dont les instructions ressemblent à des phrases en anglais.

SQL, un langage déclaratif

- ▶ SQL : décrit le résultat à obtenir, sans décrire comment l'obtenir
- ▶ Chaque SGBD : possède un interpréteur SQL qui exécute la requête (pour obtenir le résultat demandé)
- ▶ Traitement de la requête : optimisé pour accélérer les calculs
- ▶ Traitement sous-jacent : propre à chaque SGBD
- ▶ Résultats : indépendants du SGBD



Les 3 (ou 4) sous-langages de SQL



DML – Data Manipulation Language

Permet d'extraire et de mettre à jour les données contenues dans la base. La partie *extraction de données* est parfois considérée à part entière sous la dénomination Data Query Language (DQL).

DDL – Data Definition Language

Permet de créer, de modifier ou de supprimer les relations de la base de données, de définir des liens entre ces relations et leurs clés primaires.

DCL – Data Control Language

Permet de gérer des protections d'accès aux relations ou tables en environnements multi-utilisateurs.



Plan du cours



Introduction

Data Query Language

Data Manipulation Language

Data Definition Language

Data Control Language



Plan



Introduction

Data Query Language

- Requêtes de sélection simple

- Jointures

- Agrégation des valeurs d'un attribut

- Sous-requêtes

- Opérateurs ensemblistes

- Division

- Expressions et fonctions

Data Manipulation Language

Data Definition Language



Plan



Introduction

Data Query Language

- Requêtes de sélection simple

- Jointures

- Agrégation des valeurs d'un attribut

- Sous-requêtes

- Opérateurs ensemblistes

- Division

- Expressions et fonctions

Data Manipulation Language

Data Definition Language



Syntaxe de base : **SELECT** ... **FROM** ...

Définition

SELECT att1, att2, att3... **FROM** table1, table2,...

- ▶ Contrainte : les attributs att1, att2, ... _____

- ▶ Effectue le _____ les tables table1, table2,...
- ▶ _____ sur les attributs att1, att2, att3...

Exemple : nom et prénom des étudiant·e·s

ETUDIANT

Número*	Nom	Prenom	DateNaissance	Login	MotDePasse
23794	Avenia	Adrien	18-08-1998	aaavenia	7509d123bce1a...
32827	Signoret	Adrien	24-03-1998	asignoret	a23e5f3c567d0...
32911	Gigant	Simon	26-02-1998	sgigant	6b28830776de6...

SELECT Nom, Prenom **FROM** ETUDIANT

Nom	Prenom
Avenia	Adrien
Signoret	Adrien
Gigant	Simon

Exemple : nom des étudiant·e·s et modules

ETUDIANT

Número*	Nom
23794	Avenia
34812	Smolinski

MODULE

CodeModule*
BD_L1
PROG_L1

ETUDIANT *pcart* MODULE

Número*	Nom	CodeModule*
23794	Avenia	BD_L1
23794	Avenia	PROG_L1
34812	Smolinski	BD_L1
34812	Smolinski	PROG_L1

SELECT Nom, CodeModule
FROM ETUDIANT, MODULE

Nom	CodeModule*
Avenia	BD_L1
Avenia	PROG_L1
Smolinski	BD_L1
Smolinski	PROG_L1

Doublons unifiés : SELECT DISTINCT ... FROM ...

Exemple de requête _____

ETUDIANT

Numero*	Nom	Prenom	DateNaissance	Login	MotDePasse
23794	Avenia	Adrien	18-08-1998	aavenia	7509d123bce1a...
32827	Signoret	Adrien	24-03-1998	asignoret	a23e5f3c567d0...
32911	Gigant	Simon	26-02-1998	sgigant	6b28830776de6...
33818	Maillard	Cynthia	06-05-1998	cmaillard	2847b2605f6a1...
34812	Smolinski	Marie	21-11-1998	ssmolinski	f8eded6c815c7...

```
SELECT Prenom  
FROM ETUDIANT
```

Prenom

Adrien
Adrien
Simon
Cynthia
Marie

- ▶ Le résultat d'une requête SQL n'est donc pas une relation

Exemple de requête _____

ETUDIANT

Numero*	Nom	Prenom	DateNaissance	Login	MotDePasse
23794	Avenia	Adrien	18-08-1998	aavenia	7509d123bce1a...
32827	Signoret	Adrien	24-03-1998	asignoret	a23e5f3c567d0...
32911	Gigant	Simon	26-02-1998	sgigant	6b28830776de6...
33818	Maillard	Cynthia	06-05-1998	cmaillard	2847b2605f6a1...
34812	Smolinski	Marie	21-11-1998	ssmolinski	f8eded6c815c7...

```
SELECT DISTINCT Prenom  
FROM ETUDIANT
```

Prenom

Adrien
Simon
Cynthia
Marie

Renommer les col^{ne}. : **SELECT** expr **AS** Nom **FROM**...

Exemple de renommage de colonne résultat

ETUDIANT

Numero*	Nom	Prenom	DateNaissance	Login	MotDePasse
23794	Avenia	Adrien	18-08-1998	aavenia	7509d123bce1a...
32827	Signoret	Adrien	24-03-1998	asignoret	a23e5f3c567d0...
32911	Gigant	Simon	26-02-1998	sgigant	6b28830776de6...
33818	Maillard	Cynthia	06-05-1998	cmaillard	2847b2605f6a1...
34812	Smolinski	Marie	21-11-1998	ssmolinski	f8eded6c815c7...

```
SELECT DISTINCT Prenom AS FirstName  
FROM ETUDIANT
```

FirstName
Adrien
Simon
Cynthia
Marie

► Construction **AS** FirstName : _____



Tables sélectionnées dans **SELECT ... FROM ...**

Combiner plusieurs tables dans le **FROM ...**

SELECT ...

FROM Table1 [**AS** T1], Table2 [**AS** T2] ...

- ▶ **FROM...** : source de données de la requête en indiquant les tables ou les sous-requêtes qui seront utilisées
- ▶ Construction **Table AS T** : renommage de **Table** en **T**

Exemple de sélection sur plusieurs tables

ETUDIANTS1

Numero*	Nom
23794	Avenia
32911	Gigant
34812	Smolinski

MODULES1

CodeModule*
BD_L1
PROG_L1

```
SELECT Nom, CodeModule  
FROM ETUDIANTS1, MODULES1
```

Nom	CodeModule
Avenia	BD_L1
Avenia	PROG_L1
Gigant	BD_L1
Gigant	PROG_L1
Smolinski	BD_L1
Smolinski	PROG_L1

- ▶ Remarque : _____

Ordonner les résultats : ORDER BY

Définition

- ▶ Syntaxe : `SELECT...ORDER BY` expr1 [ASC/DESC], expr2 [ASC/DESC]
- ▶ `ASC` : _____
- ▶ `DESC` : _____
- ▶ `expr` : expression sur les attributs (event^t attribut seul).

Nom, prénom des étudiant·e·s, par ordre alphab. sur les noms et prénoms

```
SELECT NomEtudiant, PrenomEtudiant FROM ETUDIANT  
ORDER BY NomEtudiant ASC, PrenomEtudiant ASC
```

ETUDIANT

NumeroEtudiant*	NomEtudiant	PrenomEtudiant	DateNaissanceEtudiant	LoginEtudiant	MotDePasseEtudiant
23794	Avenia	Adrien	18-08-1998	aavenia	7509d123bce1a...
34812	Smolinski	Marie	21-11-1998	ssmolinski	f8eded6c815c7...
33818	Maillard	Cynthia	06-05-1998	cmaillard	2847b2605f6a1...

Résultat :

NomEtudiant	PrenomEtudiant
Avenia	Adrien
Maillard	Cynthia
Smolinski	Marie

Sélection conditionnelle : **SELECT...FROM...WHERE...**

Syntaxe de **WHERE...**

SELECT ... FROM ... WHERE ConditionDeSelection

- ▶ **ConditionDeSelection** : expression logique évaluée pour chaque *t*-uplet
 - ▶ comparaison entre 2 éléments : **>**, **>=**, **<=**, **<**, **=**, **<>**
 - ▶ connecteurs logiques : **...AND...**, **...OR...**, **NOT...**
 - ▶ comparaison avec **NULL** : _____, _____
 - ▶ test d'appartenance : _____, _____, _____
- ▶ dans les chaînes de caractères, utilisation de caractères génériques : **_** (remplace 1 caractère), **%** (remplace une chaîne de caractères)

Exemple : Numéro et code module des étudiant·e·s inscrit·e·s en 2016

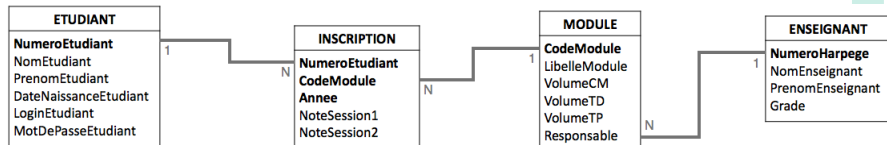
INSCRIPTION

NoEtudiant*	CodeModule	Annee	NoteSession1	NoteSession2
23794	BD_L1	2015	7	9
23794	BD_L1	2016	15	NULL
23794	PROG_L1	2015	16	NULL
32911	BD_L1	2016	7	12
32911	PROG_L1	2016	16	NULL
34812	BD_L1	2016	10	NULL

```
SELECT NoEtudiant, CodeModule  
FROM INSCRIPTION WHERE Annee = 2016
```

NoEtudiant	CodeModule
23794	BD_L1
32911	BD_L1
32911	PROG_L1
34812	BD_L1

SELECT...FROM...WHERE... : exemples 1



- ▶ Numéro des étudiant·e·s né·e·s en 1998, ou après

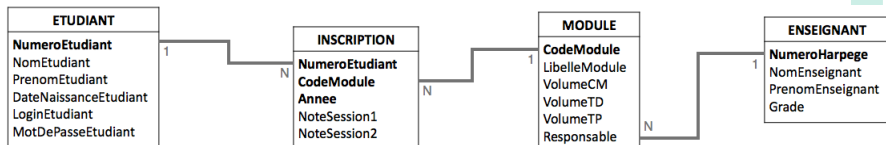
- ▶ Numéro des étudiant·e·s inscrit·e·s dans un module en 2015 ou en 2016

- ▶ Numéro des étudiant·e·s dont le prénom commence par A.

_____ 'a%' _____ 'A%'



SELECT...FROM...WHERE... : exemples 2



- ▶ Numéro des étudiant·e·s ayant été en seconde session et code du module associé.

- ▶ Numéro des étudiant·e·s ayant validé le module BD_L1 en session 1 avec une note entre 12 et 14.



Plan

Introduction

Data Query Language

Requêtes de sélection simple

Jointures

Agrégation des valeurs d'un attribut

Sous-requêtes

Opérateurs ensemblistes

Division

Expressions et fonctions

Data Manipulation Language

Data Definition Language



Jointure SQL

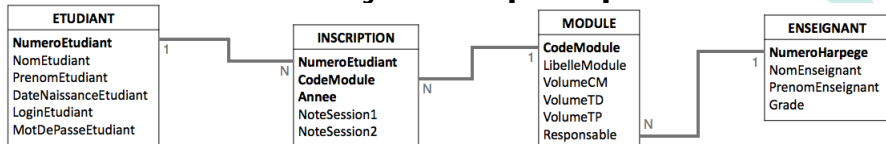
Mêmes motivation qu'en algèbre relationnelle

- ▶ Organisation relationnelle \rightsquigarrow données réparties dans plusieurs tables
- ▶ Motivation : associer les données de plusieurs tables dans une même requête.
- ▶ Fondée sur une sélection d'enregistrements _____

Plusieurs types de jointures entre 2 tables

- ▶ ... **CROSS JOIN**... : produit cartésien des 2 tables. Attention, le nombre de résultats est en général très élevé.
- ▶ ...**INNER JOIN**...**ON**... : jointure interne classique. Jointures la plus commune.
- ▶ ... **NATURAL JOIN**... : jointure interne naturelle (attributs homonymes)
- ▶ **LEFT JOIN** : jointure externe gauche.
- ▶ **RIGHT JOIN** : jointure externe droite.
- ▶ **FULL JOIN** : jointure externe totale.

Jointure interne : 2 syntaxes principales



`FROM Table1 INNER JOIN Table2 ON Table1.attA = Table2.attB`

- ▶ Exemple : libellé des modules et nom de leur responsable

```
SELECT LibelleModule, NomEnseignant
FROM MODULE INNER JOIN ENSEIGNANT
ON MODULE.Responsable = ENSEIGNANT.NoHarpege
```

- ▶ Inconvénient : _____

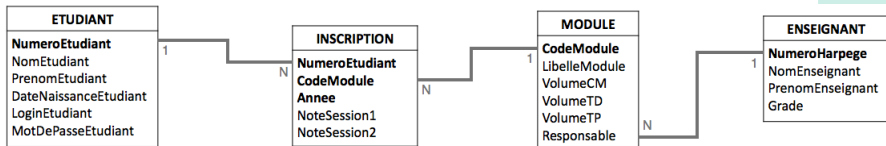
`FROM Table1, Table2 WHERE Table1.attA = Table2.attB`

- ▶ Même exemple : libellé des modules et nom de leur responsable

```
SELECT LibelleModule, NomEnseignant
FROM MODULE, ENSEIGNANT
WHERE MODULE.Responsable = ENSEIGNANT.NoHarpege
```

- ▶ Inconvénient : _____

NATURAL JOIN pour des attributs homonymes



Deux tables T1 et T2 avec des attributs homonymes

- ▶ Jointure interne d'égalité _____
- ▶ Exemples : nom et numéro des étudiant·e·s inscrit·e·s en 2016

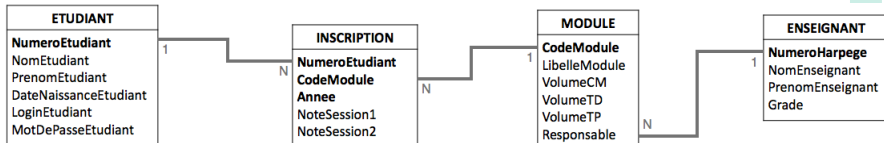
```
SELECT ETUDIANT.NomEtudiant, ETUDIANT.NumeroEtudiant
FROM ETUDIANT NATURAL JOIN INSCRIPTION
WHERE Annee = 2016
```



Auto-jointure

Définition pour une table T

Jointure d'une table T avec elle-même, mais renommées par T **AS** T prime pour pouvoir comparer les attributs de T avec ceux de T' .



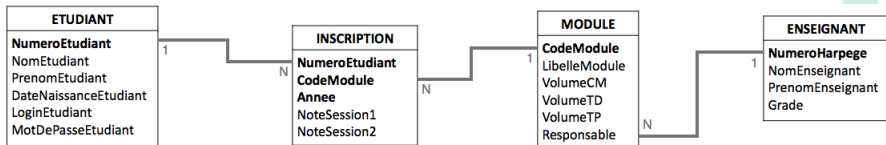
Numéro des étudiant·e·s inscrit·e·s au m module, la m année que 23974

```
SELECT INSCRIPTION.NumeroEtudiant FROM INSCRIPTION, INSCRIPTION AS I2
WHERE I2.NumeroEtudiant = 23974
AND INSCRIPTION.CodeModule=I2.CodeModule AND INSCRIPTION.Annee=I2.Annee
```

Nom, prénom des étudiant·e·s inscrit·e·s au m module, la m année que 23974

```
SELECT NomEtudiant, PrenomEtudiant
FROM ETUDIANT, INSCRIPTION, INSCRIPTION AS I2
WHERE I2.NumeroEtudiant = 23974
AND INSCRIPTION.CodeModule = I2.CodeModule AND INSCRIPTION.Annee = I2.Annee
AND ETUDIANT.NumeroEtudiant = INSCRIPTION.NumeroEtudiant
```

A vous de jouer



Nom et prénom des étudiant·e·s inscrit·e·s en 2021

Nom des étudiant·e·s ayant validé des modules sans TP en session 1

Libellé des modules qui ont eu des responsables différent·e·s au cours du temps

Traduction d'un SELECT minimal en algèbre relationnelle

```
SELECT DISTINCT A1,A2,...,An  
FROM R1,R2,...,Rm  
WHERE <CONDITION(A1,...,An,An+1,...,An+k)>
```

est équivalent à l'expression en algèbre relationnelle

```
P1    = R1 pcart R2  
...  
Pnm1 = Pnm2 pcart Rm  
SEL   = S(CONDITION(A1,...,An,An+1,...,An+k))Pnm1  
RES  = [A1, ... ,An] SEL
```



Plan

Introduction

Data Query Language

Requêtes de sélection simple

Jointures

Agrégation des valeurs d'un attribut

Sous-requêtes

Opérateurs ensemblistes

Division

Expressions et fonctions

Data Manipulation Language

Data Definition Language



Fonctions d'agrégation de valeurs

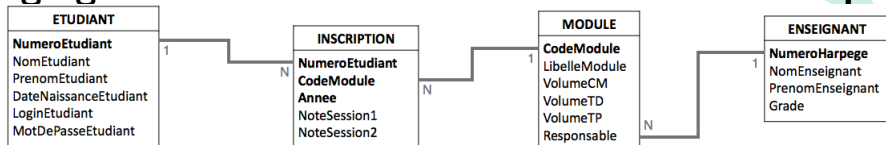
Introduction

- ▶ Objectif : construire une information _____
_____ (**MIN**, **AVG**).
- ▶ Contrainte : s'appliquent _____

Exemple de fonctions d'agrégation

- ▶ **MIN**(Att) : plus petite valeur de l'attribut Att
- ▶ **MAX**(Att) : plus grande valeur de l'attribut Att
- ▶ **SUM**(Att) : _____ de l'attribut Att
- ▶ **COUNT**(Att) : _____ de l'attribut Att
- ▶ **AVG**(Att) : _____ de l'attribut Att
- ▶ **VARIANCE**(Att) : _____ l'attribut Att
- ▶ **STDDEV**(Att) : _____ l'attribut Att

Agrégation des valeurs de ts les résultats : exemples



Date de naiss. de l'étudiant·e· le/la plus âgé·e· inscrit·e· en BD_L1 en 2023

Nombre d'étudiant·e·s inscrit·e·s au module BD_L1 en 2016

Moyenne des notes de première session au module BD_L1 en 2016

Particularités de la fonction COUNT

3 types d'appels

- ▶ **COUNT** (*) : _____
- ▶ **COUNT** (Att) : _____

- ▶ **COUNT** (**DISTINCT** Att) : _____

Exemples d'utilisation de la fonction COUNT avec la table ENSEIGNANT

NumeroHarpege*	NomEnseignant	PrenomEnseignant	Grade
7358	Janey	Nicolas	MCF
7914	Dadeau	Frédéric	MCF
22223	Guyennet	Hervé	PR
32598	Paquette	Guillaume	NULL

- ▶ **SELECT COUNT(*) FROM ENSEIGNANT** ~> _
- ▶ **SELECT COUNT(Grade) FROM ENSEIGNANT** ~> _
- ▶ **SELECT COUNT(DISTINCT Grade) FROM ENSEIGNANT** ~> _

Regroupements par **GROUP BY** et agrégation

Motivations.

- ▶ Agrégations précédentes : grossières car appliquées sur tous les résultats.
- ▶ Regrouper des données _____
- ▶ Pour faire _____

Définition

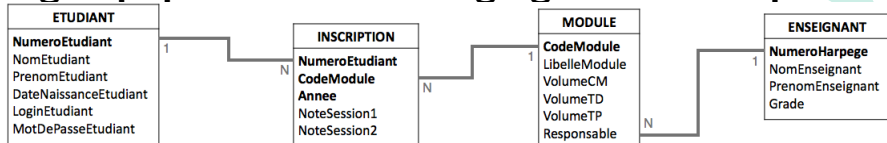
- ▶ Syntaxe : **SELECT...FROM... WHERE...GROUP BY** Att1, Att2...
- ▶ Att1, Att2... : regroup^t des enregist^t qui ont ces attributs égaux

Expl. : nombre de modules suivis par étudiant·e· par année issu d'INSCRIPTION

NumeroEtudiant*	CodeModule*	Annee*	NoteSession1	NoteSession2	NbModSuivis
23794	BD_L1	2015	7	9	2
23794	BD_L1	2016	15	NULL	1
23794	PROG_L1	2015	16	NULL	

1. Préliminaires avec **ORDER BY** :
SELECT CodeModule **FROM** INSCRIPTION **ORDER BY** NumeroEtudiant,Annee
2. Transformation **GROUP BY** :
SELECT COUNT(*) **AS** NbModSuivis **FROM** INSCRIPTION
GROUP BY NumeroEtudiant, Annee

Regroup^t par **GROUP BY** et agrégation : exemples



Nombre d'étudiants par module et par année issu d'INSCRIPTION

NumeroEtudiant*	CodeModule*	Annee*	NoteSession1	NoteSession2
23794	BD_L1	2015	7	9
23794	BD_L1	2016	15	NULL
23794	PROG_L1	2015	16	NULL
32911	BD_L1	2016	7	12

Résultat :

Annee	CodeModule	Nb
2015	BD_L1	1
2016	BD_L1	2
2015	PROG_L1	1

1. Préliminaires avec **ORDER BY** :

```
SELECT Annee, CodeModule FROM INSCRIPTION ORDER BY Annee, CodeModule
```

2. Transformation **GROUP BY** :

```
_____
_____
```

Nom des enseignant·e·s et nombre de modules \neq en responsabilité

```
_____
_____
-- NomEnseignant dans le SELECT-> _____
```

Groupe sélectionné selon une fct. d'agrégation

Instruction **HAVING**

Motivations

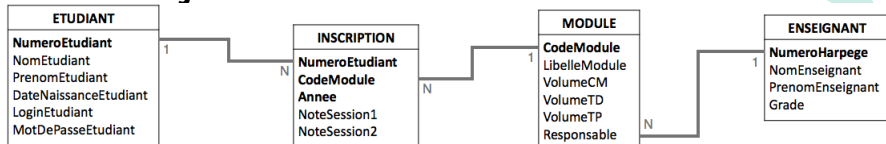
- ▶ Sélectionner un regroupement en fonction de la valeur d'une fonction d'agrégation (p.ex. étudiant·e·s dont la moyenne sur tous les modules de l'année 2023 est plus grande que 16).
- ▶ Précisé dans le **WHERE C**? _____

HAVING : définition

- ▶ Syntaxe : **SELECT...FROM...WHERE...GROUP BY...HAVING** f(Att)...
- ▶ Doit contenir un _____
- ▶ f(Att) ... : condition portant sur la fonction d'agrégation f(Att)

Nom des enseignant·e·s qui sont responsables d'au moins 4 modules \neq

A vous de jouer



Numéro des modules où le nombre d'inscrit·e·s en 2022 est supérieur à 100

Nom et prénom des étudiant·e·s qui ont subi plus de 100h de CM en 2023

Plan



Introduction

Data Query Language

Requêtes de sélection simple

Jointures

Agrégation des valeurs d'un attribut

Sous-requêtes

Opérateurs ensemblistes

Division

Expressions et fonctions

Data Manipulation Language

Data Definition Language



Sous-requête : introduction



Intuition : imbrication de requêtes

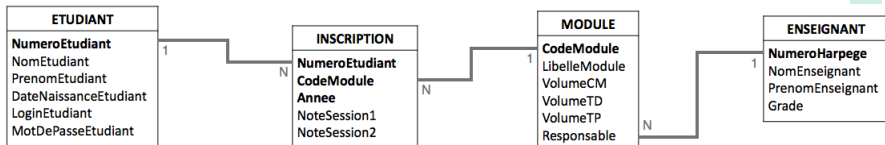
Les résultats d'une sous-requête : _____
_____ comme **IN**, **=**, ...

Code générique

```
SELECT ... FROM ... WHERE  
Attr <OPERATEUR> ( SELECT ... FROM ... WHERE ... )
```



Sous-requête : exemple motivant



Modules qui ont le même responsable que le module BD_L1

- ▶ Avec une auto-jointure :

```
SELECT M1.CodeModule FROM Module AS M1, Module AS M2
WHERE M1.Responsable = M2.Responsable AND M2.CodeModule = 'BD_L1'
```

- ▶ Avec une sous-requête :

```
SELECT CodeModule FROM Module
WHERE Responsable = (SELECT Responsable FROM Module
                     WHERE CodeModule = 'BD_L1')
```

- ▶ Code avec sous-requête : plus lisible, plus simple que celui sans.

Sous-requêtes : 2 types

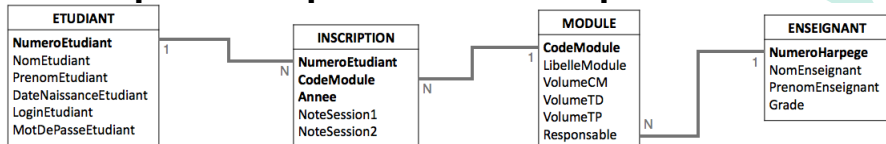


Indépendance vs. synchronisation

- ▶ Sous-requête indépendante : _____ dans la requête principale
 - ▶ \rightsquigarrow sous-requête évaluée _____ la requête principale
- ▶ Sous-requête synchronisée : _____ de la requête principale
 - ▶ \rightsquigarrow sous requête évaluée _____ la requête principale



Sous-requête indépendante : exemple



Modules qui ont le même responsable que le module BD_L1

Module

CodeModule*	LibelleModule	...	Responsable
BD_L1	Bases de données	...	7914
PROG_L1	Programmation	...	7358
TEST_M2	Test logiciel	...	7914
SR_L3	Systèmes et réseaux	...	22223
MCOO_L3	Modélisation objet	...	7914

Module

CodeModule*	LibelleModule	...	Responsable
BD_L1	Bases de données	...	7914
PROG_L1	Programmation	...	7358
TEST_M2	Test logiciel	...	7914
SR_L3	Systèmes et réseaux	...	22223
MCOO_L3	Modélisation objet	...	7914

Résultat :

CodeModule
BD_L1
TEST_M2
MCOO_L3

1. Requêtes imbriquées :

```
SELECT CodeModule FROM Module
```

```
WHERE Responsable = (SELECT Responsable FROM Module  
WHERE CodeModule = 'BD_L1')
```

Sous-requête indépendante : type du résultat

Sous-requête indépendante retournant un seul n -uplet

- ▶ Emploi des opérateurs de comparaison entre 2 n -uplets : =, <>, <, <= ...
- ▶ Exemple : étudiant·e·s qui ne sont pas “major·e·s” de BD_L1 en 2023.

```
SELECT NumeroEtudiant FROM INSCRIPTION
WHERE CodeModule = 'BD_L1' AND Annee = 2023
AND NoteSession1 < (SELECT MAX(NoteSession1) FROM INSCRIPTION
                    WHERE CodeModule = 'BD_L1' AND Annee = 2023)
```

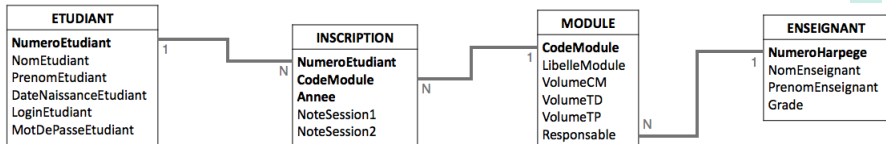
Sous-requête indép. retournant un ensemble de n -uplets

- ▶ Dans la syntaxe générique

```
SELECT ... FROM ... WHERE
Attr <OPERATEUR> ( SELECT ... FROM ... WHERE ... )
```

Emploi des opérateurs ensemblistes : IN, NOT IN, ANY, ALL ...

Sous-requête indép. ensembliste : **IN**, **NOT IN**



Numéros des étudiant·e·s qui ont côtoyé l'étudiant 23794 en 2022

```
SELECT NumeroEtudiant FROM INSCRIPTION WHERE Annee = 2022
AND CodeModule IN (SELECT CodeModule FROM INSCRIPTION
WHERE Annee = 2022 AND NumeroEtudiant = 23794)
```

Numéro des étudiant·e·s qui n'ont jamais eu l'enseignant 7914

```
SELECT NumeroEtudiant FROM ETUDIANT WHERE
NumeroEtudiant NOT IN (SELECT NumeroEtudiant FROM INSCRIPTION
NATURAL JOIN MODULE WHERE Responsable = 7914)
```



Sous-requête indép. ensembliste : ANY, ALL

Definition

- ▶ Dans la syntaxe générique

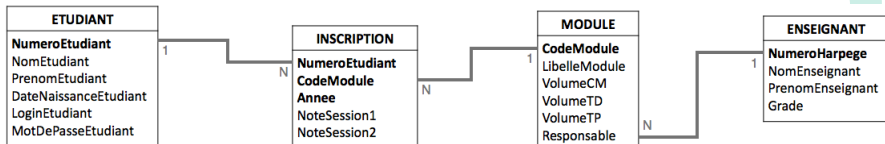
```
SELECT ... FROM ... WHERE
```

```
Attr <OPERATEUR> (SELECT ... FROM ... WHERE ...)
```

<OPERATEUR> est composé

- ▶ d'un opérateur op de comparaison entre 2 n -uplets :
=, <>, <, <= ...
- ▶ suivi de ANY ou ALL
- ▶ op ANY (SELECT ... FROM ... WHERE ...) : vraie si la condition utilisant op est vraie pour _____
 - ▶ Rem. : comparaison fautive si la sous-requête renvoie \emptyset
- ▶ op ALL (SELECT ... FROM ... WHERE ...) : vraie si la condition utilisant op est vraie pour _____
 - ▶ Rem. : comparaison vraie si la sous-requête renvoie \emptyset

Sous-req. indép. ensembliste : exemples ANY, ALL



Numéros des étudiant·e·s inscrit·e·s dans un des modules de l'enseignante 7914

```
SELECT NumeroEtudiant FROM INSCRIPTION
WHERE CodeModule = ANY (SELECT CodeModule FROM MODULE
                        WHERE Responsable = 7914)
```

Enseignant·e·s qui totalisent le plus grand nombre de responsabilités

```
SELECT NumeroHarpege FROM ENSEIGNANT, MODULE
WHERE NumeroHarpege = Responsable GROUP BY NumeroHarpege
HAVING COUNT(CodeModule) >= ALL (SELECT COUNT(CodeModule) FROM MODULE
                                GROUP BY Responsable)
```



Sous-requêtes synchronisées : définition



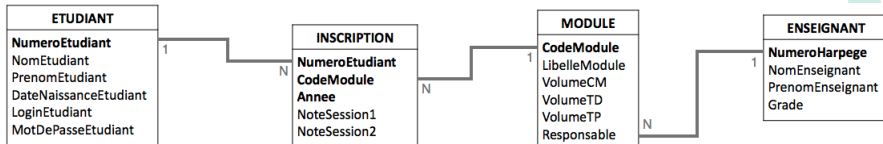
Définition

- ▶ Sous requête qui fait référence _____

- ▶ Nécessite la réévaluation de la sous-requête pour chaque n -uplet résultat de la requête principale



Sous-requêtes synchronisées : exemple motivant



Numéro des étudiant·e-s dont la note en session 1 dans un des modules est supérieure à la moyenne de celui-ci

- Pour chaque (NumeroEtudiant, CodeModule, ..., NoteSession1, ...) d'INSCRIPTION : _____

- Requête globale :

```
SELECT DISTINCT NumeroEtudiant FROM INSCRIPTION AS INSCR
WHERE NoteSession1 > (SELECT AVG(NoteSession1) FROM INSCRIPTION
                      WHERE CodeModule = INSCR.CodeModule)
```

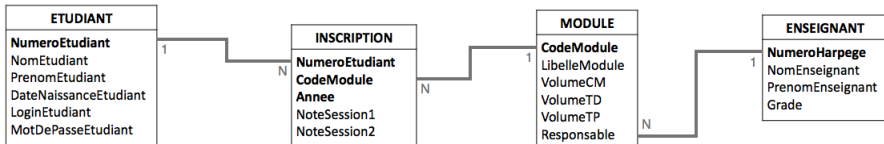
- Sous-requête : _____



Sous-requêtes synchronisées avec EXISTS

Définition

- ▶ Syntaxe : `SELECT ... WHERE [NOT] EXISTS` (Sous-requête synchronisée)
- ▶ `EXISTS` : vraie si la sous-requête **synchronisée** renvoie au moins un résultat

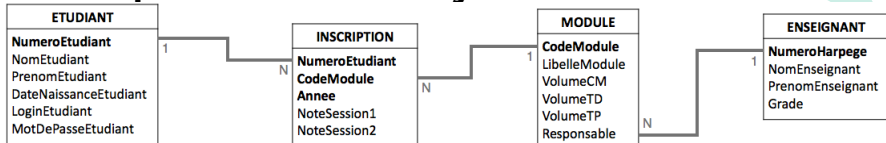


Numéros des étudiant·e·s qui ne se sont inscrit·e·s à aucun module en 2022

```
SELECT NumeroEtudiant FROM ETUDIANT
WHERE NOT EXISTS (SELECT * FROM INSCRIPTION
                  WHERE INSCRIPTION.Anee = 2022
                  AND INSCRIPTION.NumeroEtudiant = ETUDIANT.NumeroEtudiant)
```



Sous requêtes : à vous de jouer



Étudiant·e·s major·e·s du module de code 'BDL1' en première session de tous les temps

▶ _____

▶ _____

Numéro des étudiant·e·s qui ont tout validé en session 1 en 2022

▶ _____

▶ _____

Plan

Introduction

Data Query Language

Requêtes de sélection simple

Jointures

Agrégation des valeurs d'un attribut

Sous-requêtes

Opérateurs ensemblistes

Division

Expressions et fonctions

Data Manipulation Language

Data Definition Language



Opérateurs ensemblistes : définition, syntaxe

Définition

SQL propose l'utilisation des 3 opérateurs ensemblistes classiques pour composer le résultat de deux requêtes `SELECT`

- ▶ union : _____
- ▶ intersection : _____, _____
- ▶ différence : _____, _____

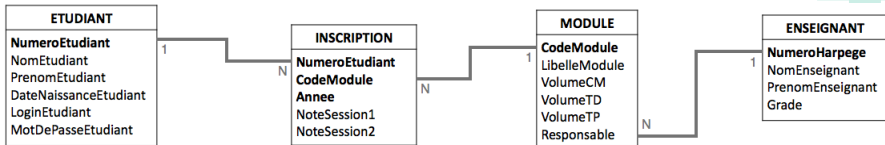
Syntaxe

```
SELECT ... FROM ...  
<Opérateur Ensembliste>  
SELECT ... FROM ...
```

Remarques

- ▶ Comme en algèbre relationnelle : même en-tête dans des 2 requêtes
- ▶ Opérateurs ensemblistes : doublons supprimés

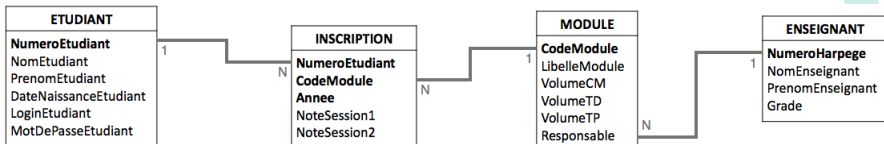
Exemple avec UNION



Noms, prénoms des enseignants MCF ou resp. d'un module sans TD

```
SELECT NomEnseignant, PrenomEnseignant FROM ENSEIGNANT WHERE Grade='MCF'  
UNION  
SELECT NomEnseignant, PrenomEnseignant FROM ENSEIGNANT, MODULE  
WHERE NumeroHarpage = Responsable AND VolumeTD = 0
```

Opérateurs ensemblistes : expl. avec INTERSECT



Noms, prénoms des enseignant·e·s MCF et resp. d'un module sans TD

```
SELECT NomEnseignant, PrenomEnseignant FROM ENSEIGNANT WHERE Grade='MCF'  
INTERSECT
```

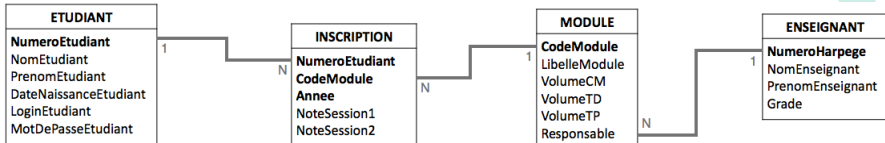
```
SELECT NomEnseignant, PrenomEnseignant FROM ENSEIGNANT INNER JOIN MODULE  
ON NumeroHarpage = Responsable WHERE VolumeTD = 0
```

Remarque : INTERSECT introduit en MySQL 8.0.31



```
SELECT NomEnseignant, PrenomEnseignant FROM ENSEIGNANT  
WHERE Grade='MCF' AND NumeroHarpege IN  
(SELECT Responsable FROM MODULE WHERE VolumeTD=0)
```

Exemple avec EXCEPT



Noms, prénoms des enseignant·e·s MCF, pas resp. d'un module sans TD

```
SELECT NumeroHarpege, NomEnseignant, PrenomEnseignant FROM ENSEIGNANT WHERE Grade = 'MCF' EXCEPT
```

```
SELECT NumeroHarpege, NomEnseignant, PrenomEnseignant FROM ENSEIGNANT INNER JOIN MODULE ON NumeroHarpege = Responsable WHERE VolumeTD = 0
```

Remarque : EXCEPT introduit en MySQL 8.0.31

```
SELECT NomEnseignant, PrenomEnseignant FROM ENSEIGNANT WHERE Grade='MCF' AND NumeroHarpege NOT IN (SELECT NumeroHarpege FROM ENSEIGNANT INNER JOIN MODULE ON NumeroHarpege = Responsable WHERE VolumeTD = 0)
```

Plan



Introduction

Data Query Language

Requêtes de sélection simple

Jointures

Agrégation des valeurs d'un attribut

Sous-requêtes

Opérateurs ensemblistes

Division

Expressions et fonctions

Data Manipulation Language

Data Definition Language



La division : rappels et définition en logique

Division de $R_1 \subset A_1 \times \dots \times A_m \times B_1 \times \dots \times B_n$ par $R_2 \subset B_1 \times \dots \times B_n$

- ▶ La plus grande table Q dont le produit $Q \times R_2$ est inclus dans R_1 .

$$\begin{array}{c} R_1 \\ \dots \\ \hline Q \end{array} \left| \begin{array}{c} R_2 \\ \dots \\ \hline R \end{array} \right. \text{ avec } R_1 = Q \times R_2 \text{ union } R$$

Exemple : Numéros des étudiant·e·s inscrit·e·s dans tous les modules

NoEtudiant*	CodeModule*
23794	PROG_L1
32911	BD_L1
32911	PROG_L1
32911	TEST_L3
34812	PROG_L1
34812	BD_L1
34812	TEST_L3

INSCRIPTION =

MOD1
CodeModule*
BD_L1
PROG_L1
TEST_L3

INSCRIPTION DIV MOD1
NoEtudiant*
32911
34812

Traduction ensembliste

$$Q = \{(a_1, \dots, a_m) \mid (a_1, \dots, a_m) \in [A_1 \dots, A_m]R_1 \text{ et} \\ \forall (b_1, \dots, b_n) \in R_2 \text{ alors } (a_1, \dots, a_m, b_1, \dots, b_n) \in R_1\}$$

Division : 2 sous-req. NOT EXISTS imbriquées

Division de $R_1 \subset A_1 \times \dots \times A_m \times B_1 \times \dots \times B_n$ par $R_2 \subset B_1 \times \dots \times B_n$

$$Q = \{(a_1, \dots, a_m) \mid (a_1, \dots, a_m) \in [A_1 \dots, A_m]R_1 \text{ et} \\ \forall (b_1, \dots, b_n). (b_1, \dots, b_n) \in R_2 \Rightarrow (a_1, \dots, a_m, b_1, \dots, b_n) \in R_1\}$$
$$\Leftrightarrow$$

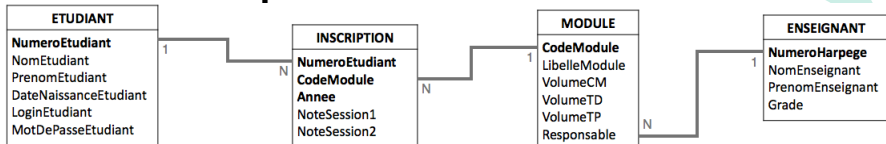
$$Q = \{(a_1, \dots, a_m) \mid (a_1, \dots, a_m) \in [A_1 \dots, A_m]R_1 \text{ et} \\ \neg (\exists (b_1, \dots, b_n). (b_1, \dots, b_n) \in R_2 \text{ et } (a_1, \dots, a_m, b_1, \dots, b_n) \notin R_1)\}$$
$$\Leftrightarrow$$

$$Q = \{(a_1, \dots, a_m) \mid (a_1, \dots, a_m) \in [A_1 \dots, A_m]R_1 \text{ et} \\ \nexists (b_1, \dots, b_n). (b_1, \dots, b_n) \in R_2 \text{ et } (a_1, \dots, a_m, b_1, \dots, b_n) \notin R_1\}$$

Traduction SQL \rightsquigarrow formule à particulariser

```
SELECT A1, ..., Am FROM R1 AS T1 WHERE
  NOT EXISTS (SELECT B1, ..., Bn FROM R2 AS T2 WHERE
    NOT EXISTS (SELECT A1, ..., Am, B1, ..., Bn FROM R1 AS T1P
      WHERE T1.A1 = T1P.A1 AND ... T1.Am = T1P.Am AND
            T2.B1 = T1P.B1 AND ... T2.Bn = T1P.Bn))
```

Division : exemple



Formule à particulariser

```
SELECT A1, ..., Am FROM R1 AS T1 WHERE
  NOT EXISTS (SELECT B1, ..., Bn FROM R2 AS T2 WHERE
    NOT EXISTS (SELECT A1, ..., Am, B1, ..., Bn FROM R1 AS T1P
      WHERE T1.A1 = T1P.A1 AND ... T1.Am = T1P.Am AND
        T2.B1 = T1P.B1 AND ... T2.Bn = T1P.Bn))
```

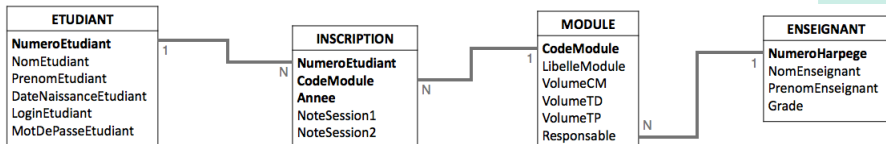
Numéros des étudiant·e·s inscrit·e·s dans tous les modules

► R1 _____

► R2 _____

```
SELECT _____ FROM _____ AS T1 WHERE
  NOT EXISTS (SELECT _____ FROM MODULE AS T2 WHERE
    NOT EXISTS (SELECT _____, _____ FROM _____ AS T1P
      WHERE T1._____ = T1P._____ AND T2._____
        = T1P._____))
```

Division : à vous de jouer



Numéros des étudiant·e·s inscrits toutes les années entre 2012 et 2017

```
SELECT NumeroEtudiant FROM INSCRIPTION AS T1 WHERE
  NOT EXISTS (SELECT Annee FROM INSCRIPTION AS T2 WHERE
    T2.Annee BETWEEN 2012 AND 2017 AND
    NOT EXISTS (SELECT NumeroEtudiant,Annee FROM INSCRIPTION AS T1P
      WHERE T1.NumeroEtudiant = T1P.NumeroEtudiant AND
      T2.Annee = T1P.Annee))
```


Plan

Introduction

Data Query Language

Requêtes de sélection simple

Jointures

Agrégation des valeurs d'un attribut

Sous-requêtes

Opérateurs ensemblistes

Division

Expressions et fonctions

Data Manipulation Language

Data Definition Language





Rappel sur les fonctions de groupe

- ▶ Applicables _____

Expressions et fonctions

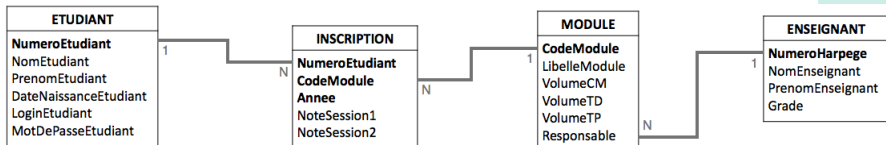
- ▶ Applicables _____

Où utiliser ces fonctions et expressions ?

A utiliser dans des instructions : **SELECT**, **WHERE**, **ORDER BY**, **GROUP BY**.



Fonctions et expressions : exemple avec **SELECT**



Moyenne du temps passé dans un module sur 14 semaines

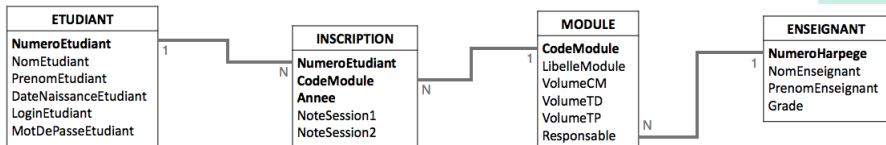
MODULE

CodeModule	LibelleModule	CM	TD	TP	...
BD_L1	Bases de données	18	18	18	...
PROG_L1	Programmation	15	21	24	...
ANG_L1	Anglais	0	18	0	..

RESULTAT

LibelleModule	TempsMoyen
Bases de données	3.86
Programmation	4.29
Anglais	1.29

Fonctions et expressions : exemple avec WHERE



Modules dont le volume horaire supérieur à 40 heures par semestre

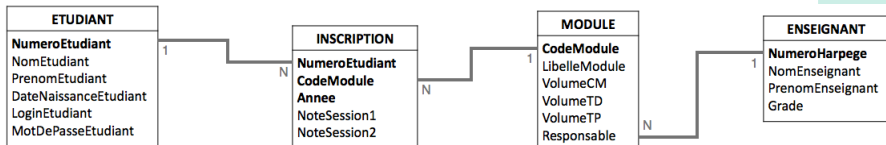
MODULE

CodeModule	LibelleModule	CM	TD	TP	...
BD_L1	Bases de données	18	18	18	...
PROG_L1	Programmation	15	21	24	...
ANG_L1	Anglais	0	18	0	..

RESULTAT

Code	LibelleModule
BD_L1	Bases de données
PROG_L1	Programmation

Fonctions et expressions : expl. avec ORDER BY

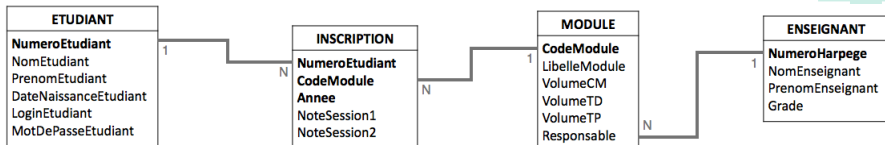


Tri des modules par ordre décroissant de volume horaire global

MODULE						
CodeModule	LibelleModule	CM	TD	TP	...	
BD_L1	Bases de données	18	18	18	...	
PROG_L1	Programmation	15	21	24	...	
ANG_L1	Anglais	0	18	0	..	

RESULTAT	
Code	LibelleModule
PROG_L1	Programmation
BD_L1	Bases de données
ANG_L1	Anglais

Fonctions et expressions : expl. avec **GROUP BY**



Moyenne par module en fonction de l'année de naissance

```
SELECT CodeModule, YEAR(DateNaissanceEtudiant) AS ANNEE,  
       AVG(NoteSession1) AS Moyenne  
FROM INSCRIPTION NATURAL JOIN ETUDIANT  
GROUP BY CodeModule, YEAR(DateNaissanceEtudiant)
```



Types Fonctions et expressions



Classement des différentes fonctions

- ▶ Fonctions numériques
 - ▶ valeur absolue **ABS()**, puissance **POW()**,...
- ▶ Fonctions sur les chaînes de caractères
 - ▶ longueur d'une chaîne **LENGTH()**, ...
 - ▶ transformation en majuscules **UPPER()**, ...
 - ▶ extraction d'une sous-chaîne **SUBSTRING()**,...
- ▶ Fonctions sur les dates
 - ▶ addition/soustraction de dates **DATEDIFF()**, des jours à une date **DATE_ADD(expr, days)**...
 - ▶ fonctions retournant la date du système **CURDATE()**, le jour de la semaine **DAYNAME(date)**,...

