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 : 04 mars de 09h30 à 11h00

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'ANSI [1]
  - 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

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...

```
SELECT Nom, Prenom FROM ETUDIANT
```

Nom	Prenom
Avenia	Adrien
Signoret	Adrien
Gigant	Simon

Exemple : nom des étudiant·e·s et modules

ETUDIANT

Numero*	Nom
23794	Avenia
34812	Smolinski

MODULE

CodeModule*
BD_L1
PROG_L1

ETUDIANT `pcart` MODULE

Numero*	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

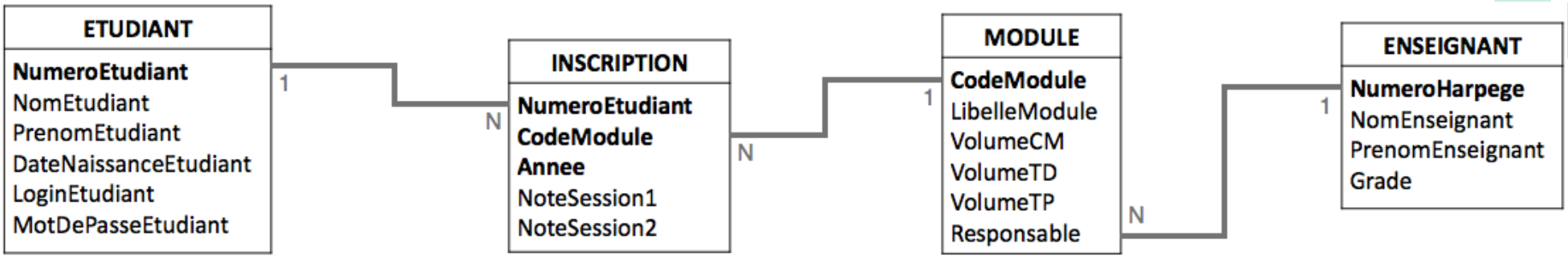
INSCRIPTION

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

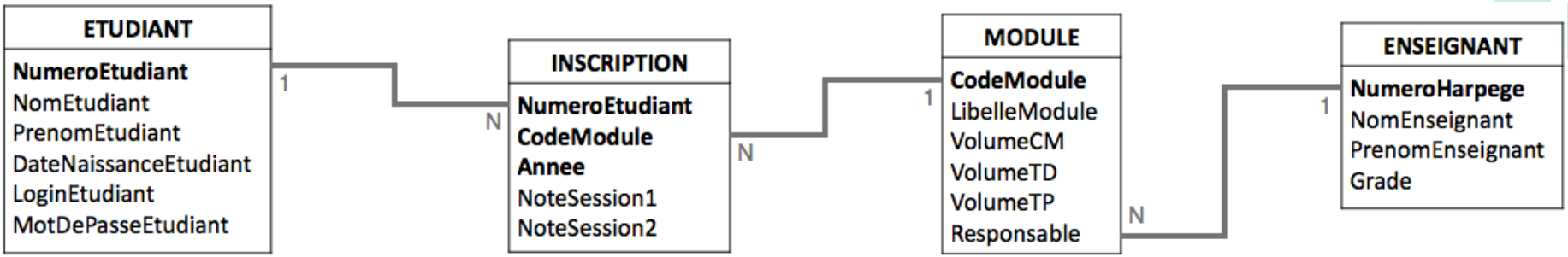


- Numéro des étudiant·e·s né·e·s en 1998, ou après

- 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.

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



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

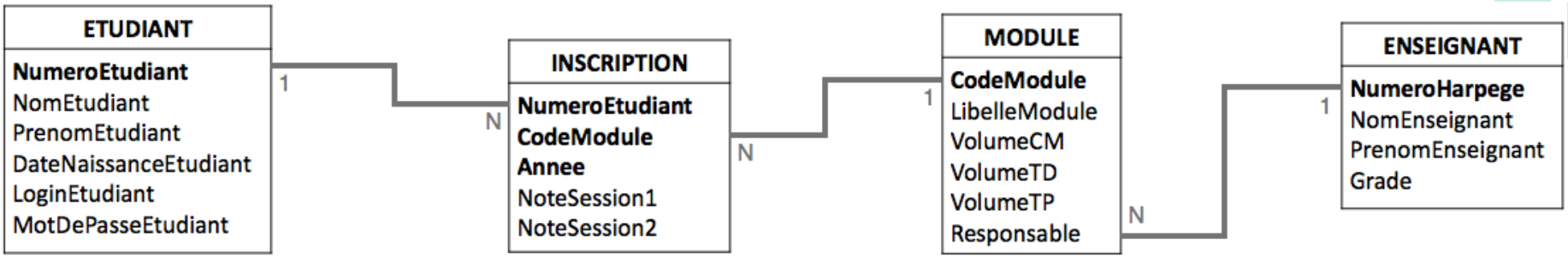
  _____

  _____

- 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.

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



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

  _____

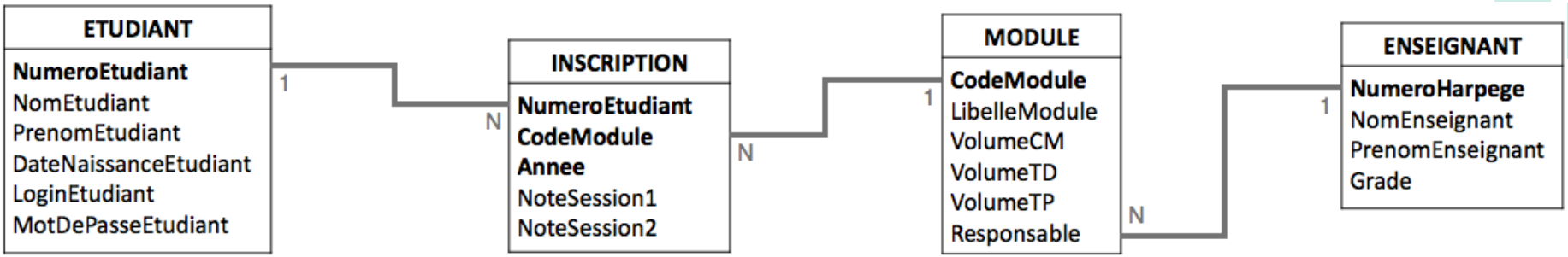
  _____

- 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.

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

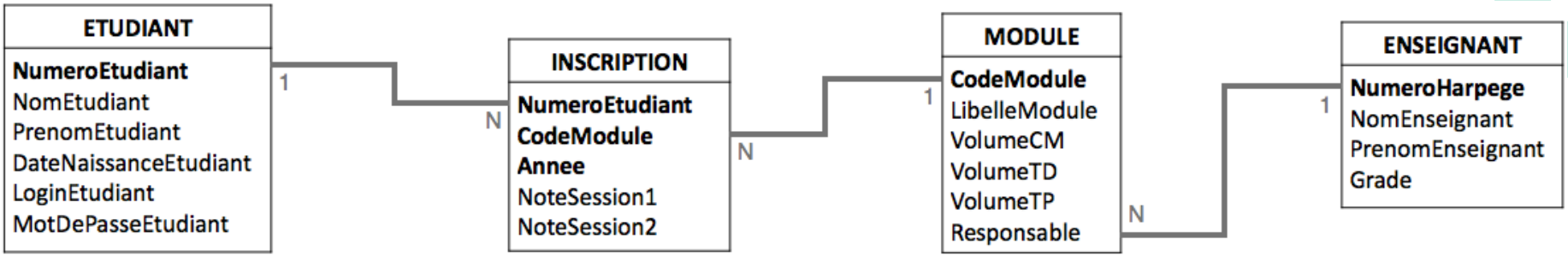
  _____

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

  _____

  _____ 'a%' _____ 'A%'

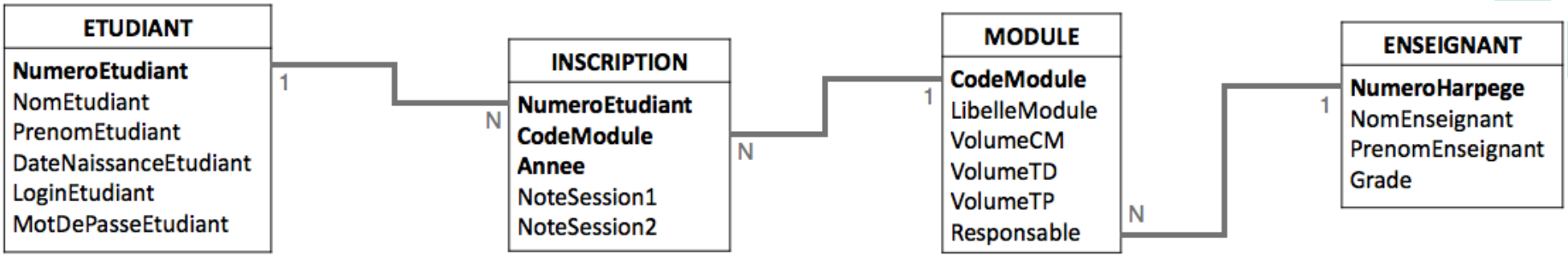
SELECT...FROM...WHERE... : exemples 2



- 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.

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



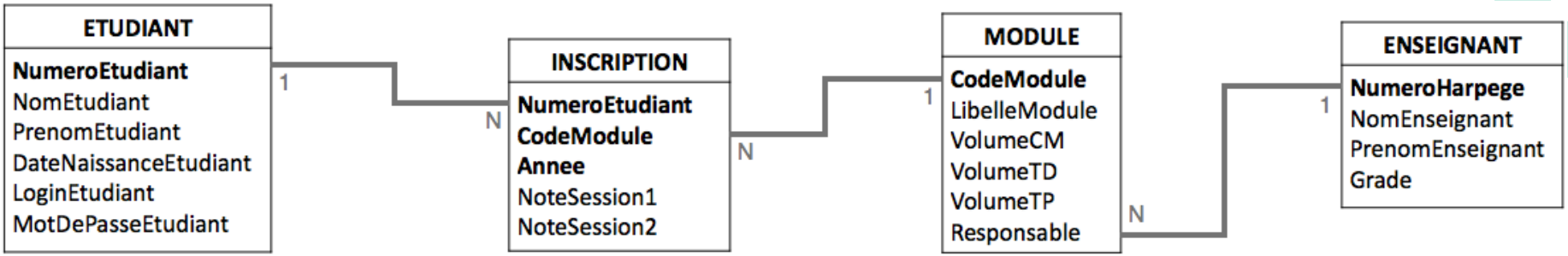
- Numéro des étudiant·e·s ayant été en seconde session et code du module associé.

  _____

  _____

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

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

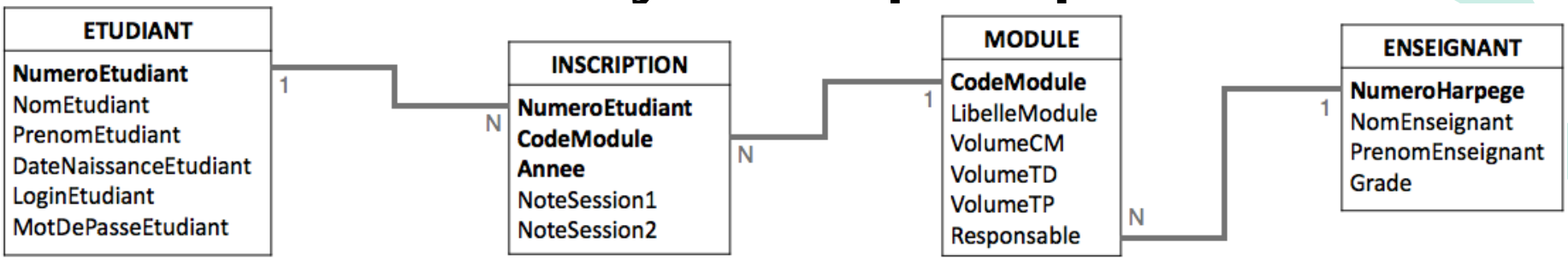
Mêmes motivation qu'en algèbre relationnelle

- 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 Responsable = NoHarpege
```

- Inconvénient : _____

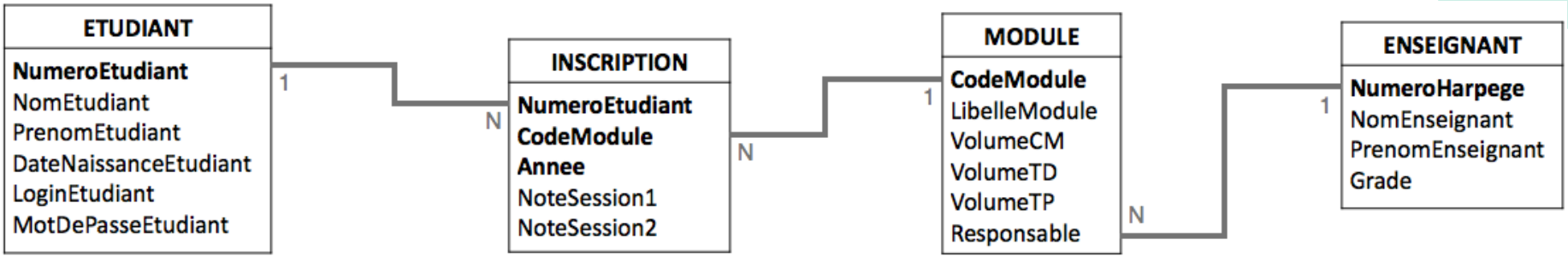
```
FROM Table1, Table2 WHERE Table1.attA = Table2.attB
```

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

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

- Inconvénient : _____

NATURAL JOIN pour des attributs homonymes



Deux tables T1 et T2 avec des attributs homonymes

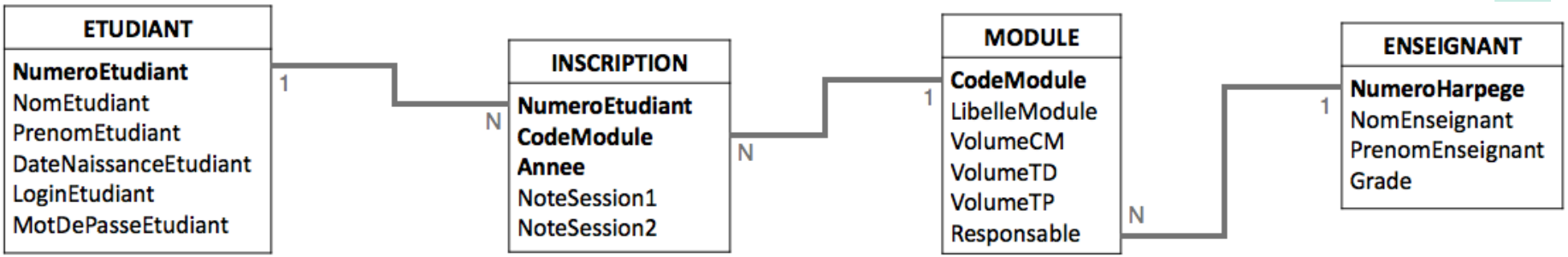
- Jointure interne d'égalité
- Exemples : nom et numéro des étudiant·e·s inscrit·e·s en 2016

```
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 Tprime` pour pouvoir comparer les attributs de T avec ceux de T'.



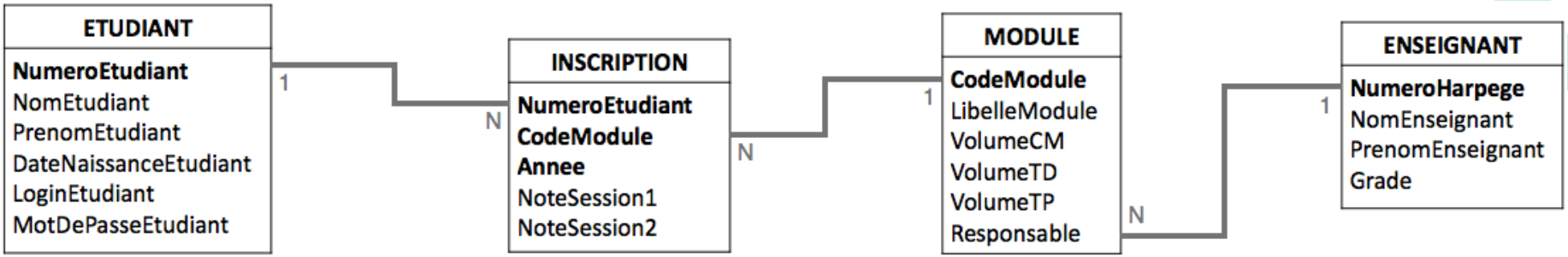
Numéro des étudiant·e·s inscrit·e·s au m̂ module, la m̂ année que 23974

```
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 m̂ module, la m̂ 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

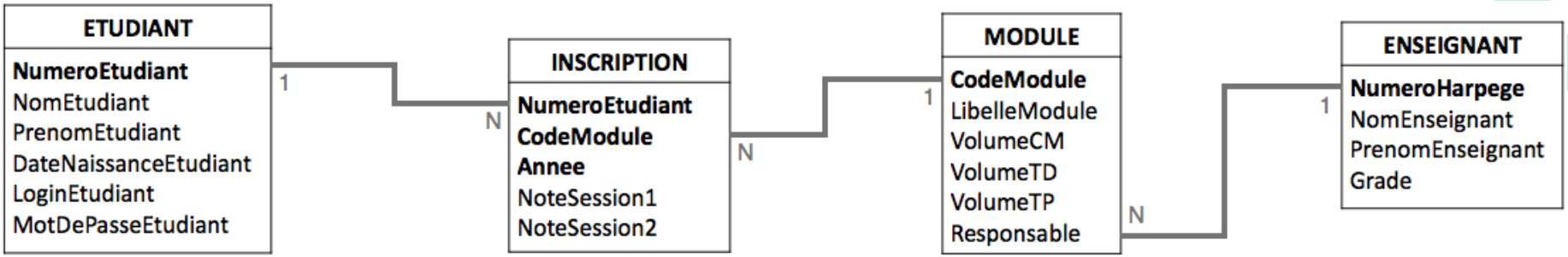


Nom et prénom des étudiant·e·s inscrit·e·s en 2021

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

A vous de jouer

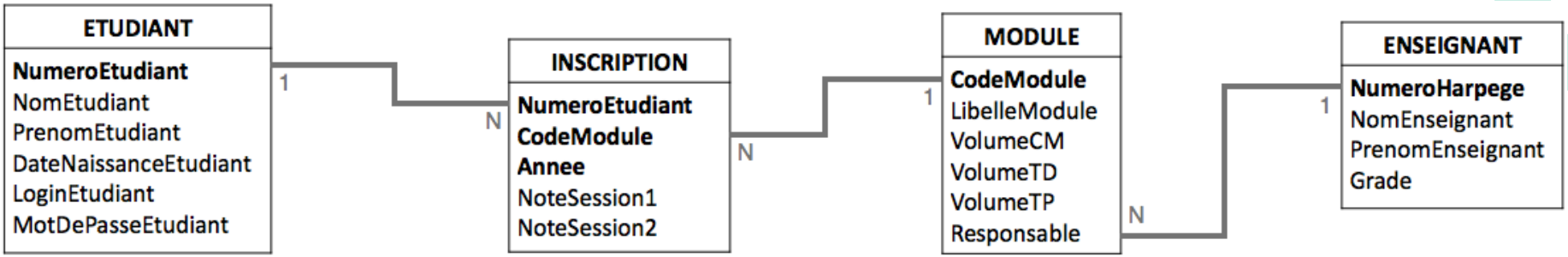


Nom et prénom des étudiant·e·s inscrit·e·s en 2021

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

A vous de jouer

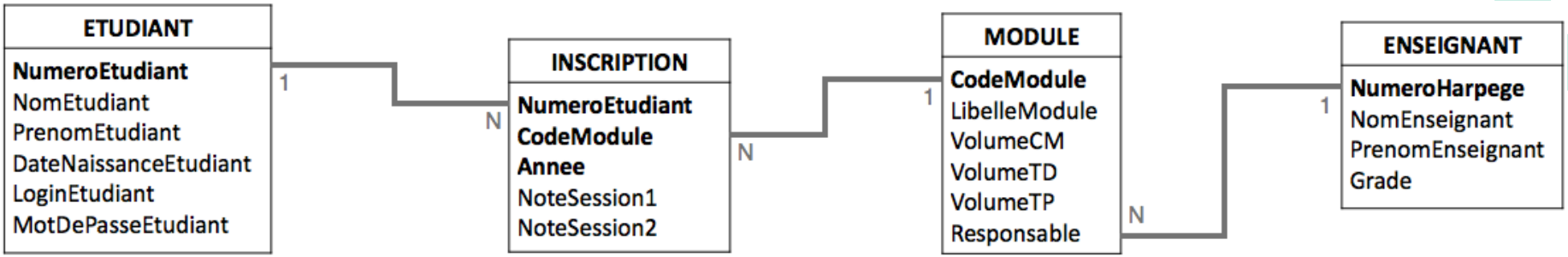


Nom et prénom des étudiant·e·s inscrit·e·s en 2021

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

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

Lien entre SELECT et algèbre relationnelle

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
...
Pmm1 = 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

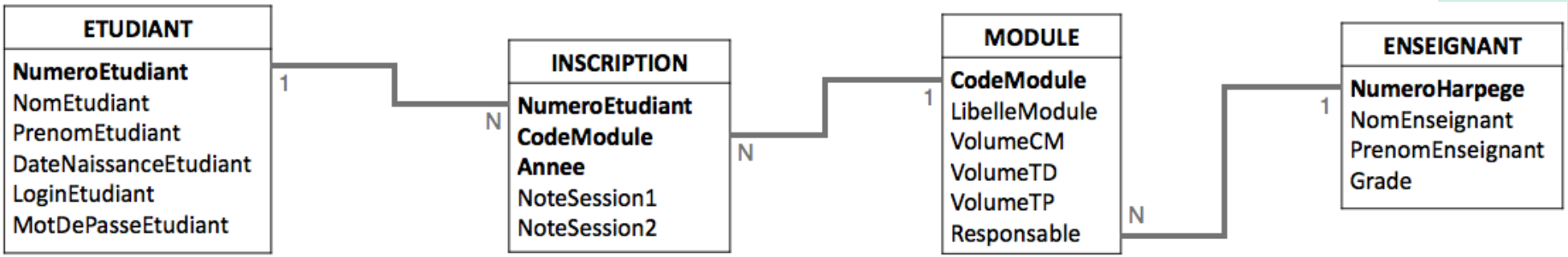
Introduction

- 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

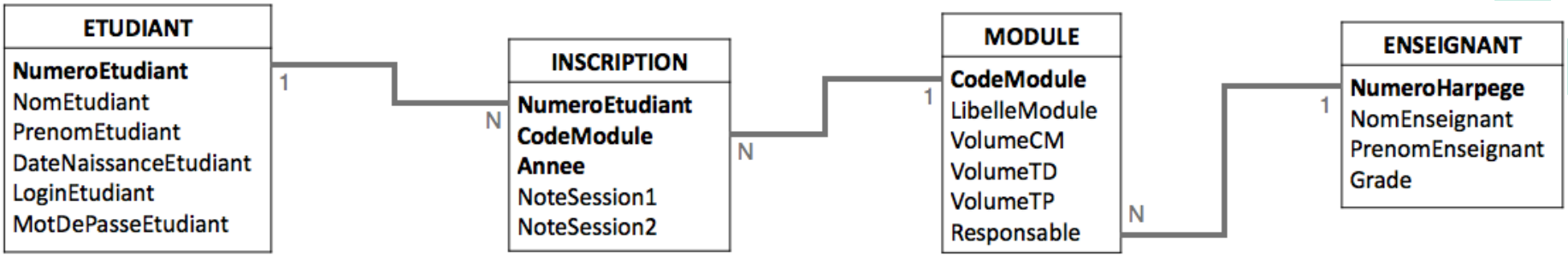


Date de naiss. de l'étudiant·e· le.la plus âgé·e· inscrit·e· en BD_L1 en 2023

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

Agrégation des valeurs de ts les résultats : exemples

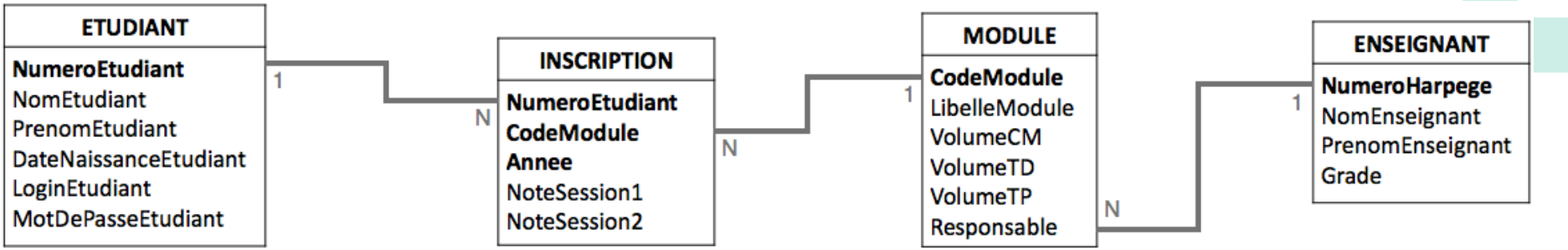


Date de naiss. de l'étudiant·e· le.la plus âgé·e· inscrit·e· en BD_L1 en 2023

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

Agrégation des valeurs de ts les résultats : exemples

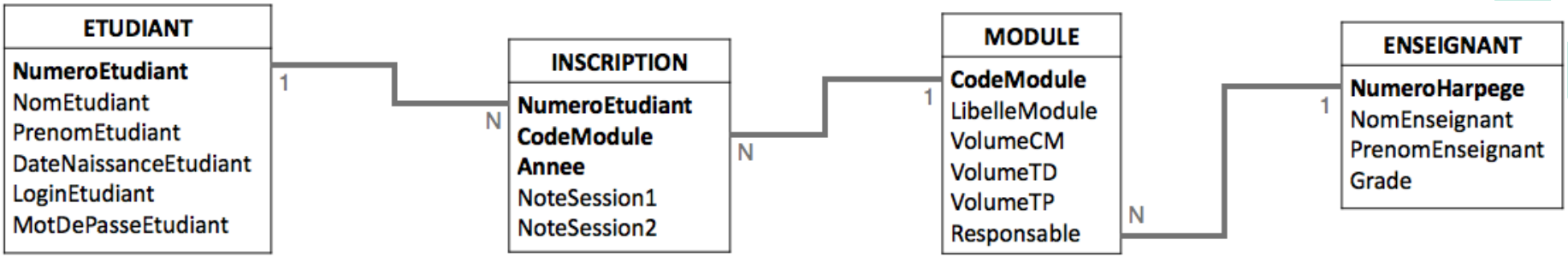


Date de naiss. de l'étudiant·e· le.la plus âgé·e· inscrit·e· en BD_L1 en 2023

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

Agrégation des valeurs de ts les résultats : exemples

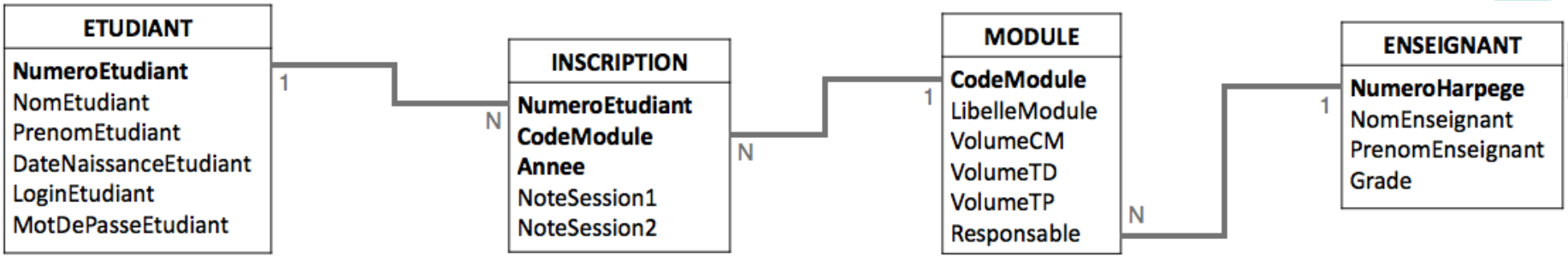


Date de naiss. de l'étudiant·e· le.la plus âgé·e· inscrit·e· en BD_L1 en 2023

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

Agrégation des valeurs de ts les résultats : exemples

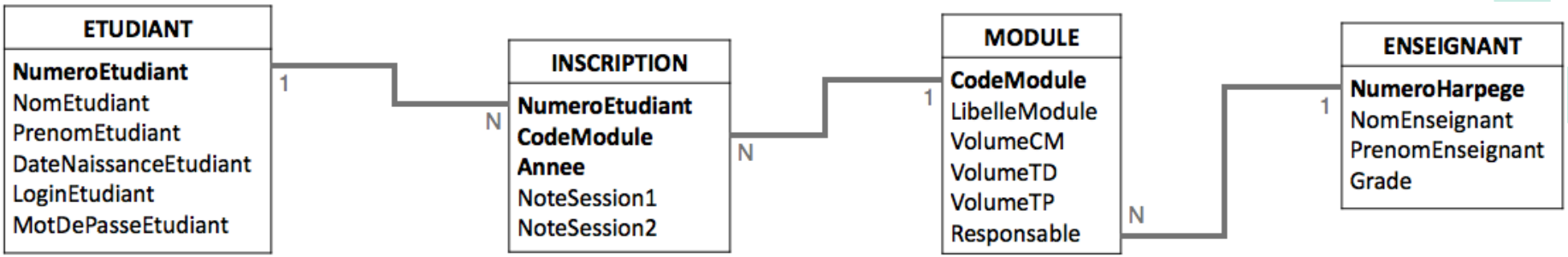


Date de naiss. de l'étudiant·e· le.la plus âgé·e· inscrit·e· en BD_L1 en 2023

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

Agrégation des valeurs de ts les résultats : exemples

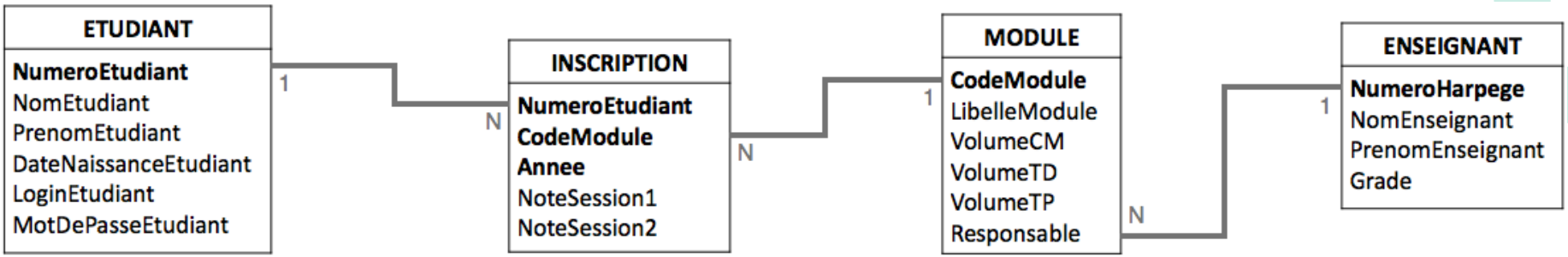


Date de naiss. de l'étudiant·e· le.la plus âgé·e· inscrit·e· en BD_L1 en 2023

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

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` ⇝

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` ⇝

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` ⇝

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 enregistr^{t} qui ont ces attributs égaux

Expl. : nombre de modules suivis par étudiant·e· 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

NbModSuivis
2
1

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 enregistr^{t} qui ont ces attributs égaux

Expl. : nombre de modules suivis par étudiant·e· 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

NbModSuivis
2
1

1. Préliminaires avec `ORDER BY` :
   `SELECT CodeModule FROM INSCRIPTION ORDER BY NumeroEtudiant,Annee`

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 enregistr^{t} qui ont ces attributs égaux

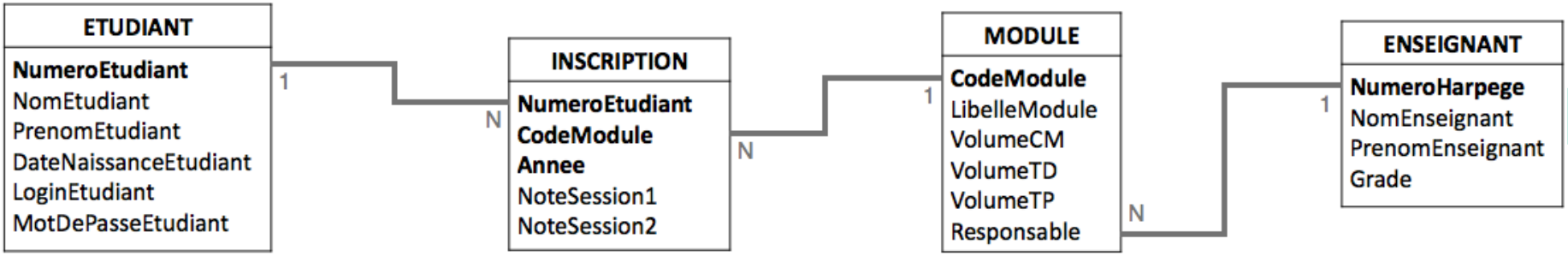
Expl. : nombre de modules suivis par étudiant·e· par année issu d'`INSCRIPTION`

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

NbModSuivis
2
1

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
   ```

Regroupt 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 :

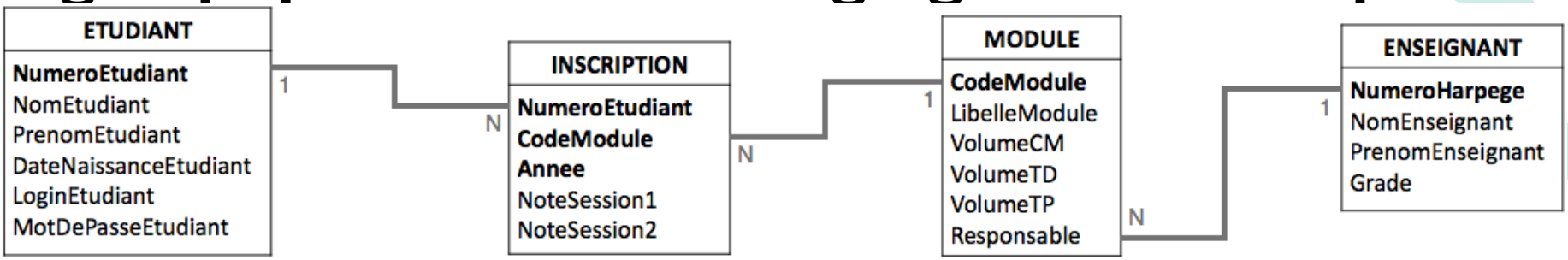
Annee	CodeModule	Nb
2015	BD_L1	1
2016	BD_L1	2
2015	PROG_L1	1

1. Préliminaires avec ORDER BY :

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

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

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)`

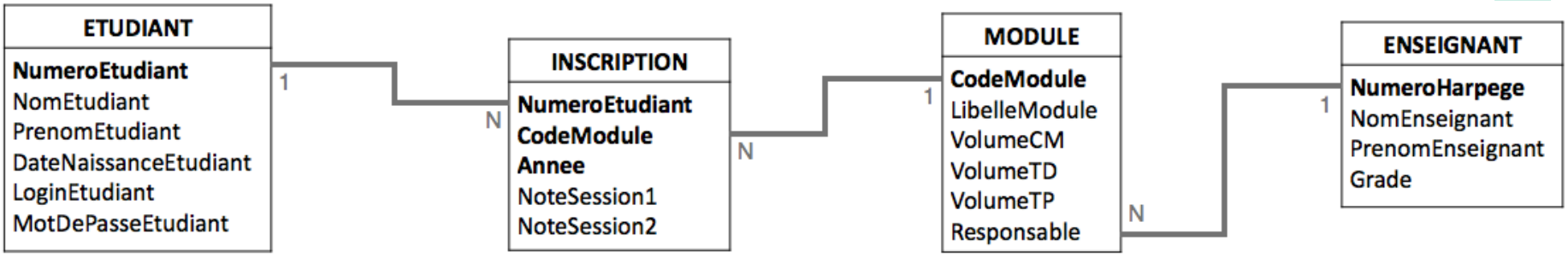
Nom des enseignant·e·s qui sont responsables d'au moins 4 modules $\neq$

_____

_____

_____

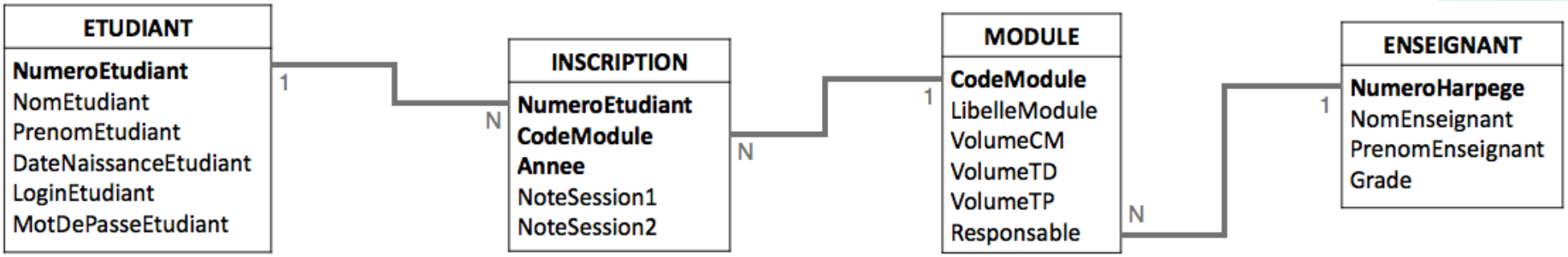
A vous de jouer



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

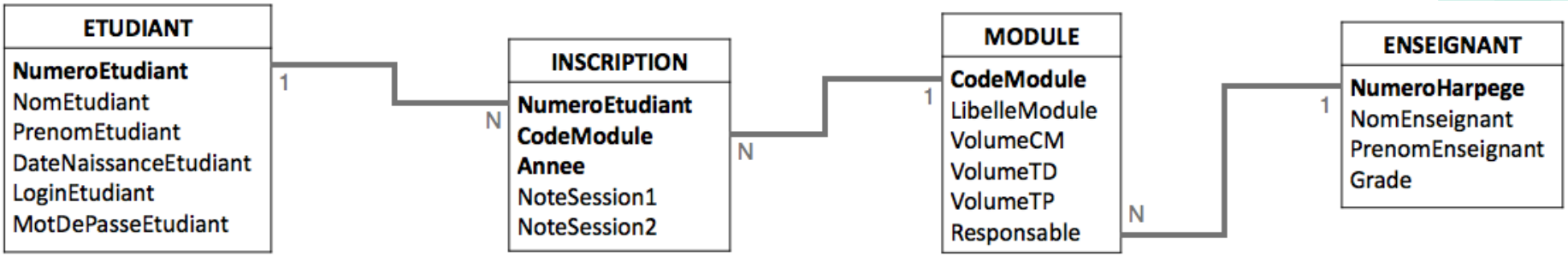
A vous de jouer



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

A vous de jouer



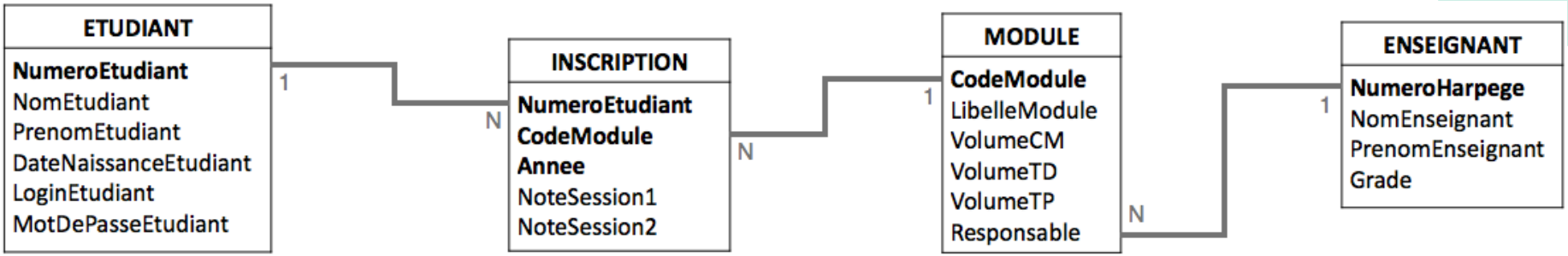
Numéro des modules où le nombre d'inscrit·e·s en 2022 est supérieur à 100

_____

_____

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

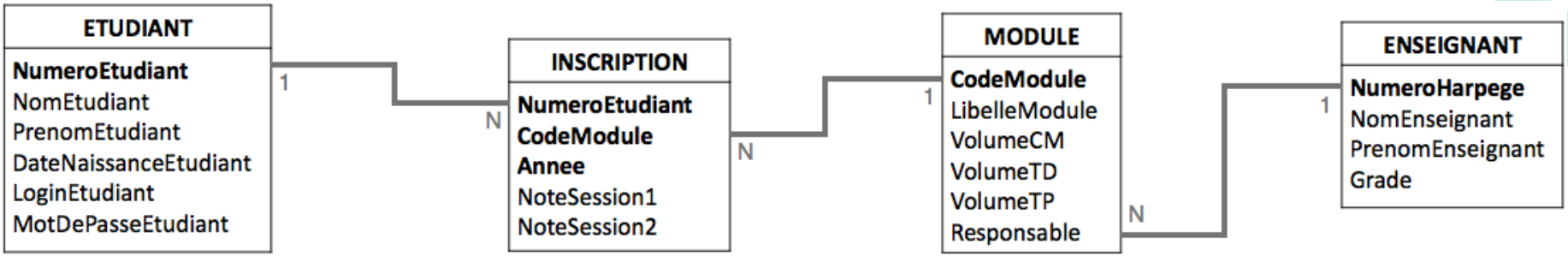
A vous de jouer



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

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

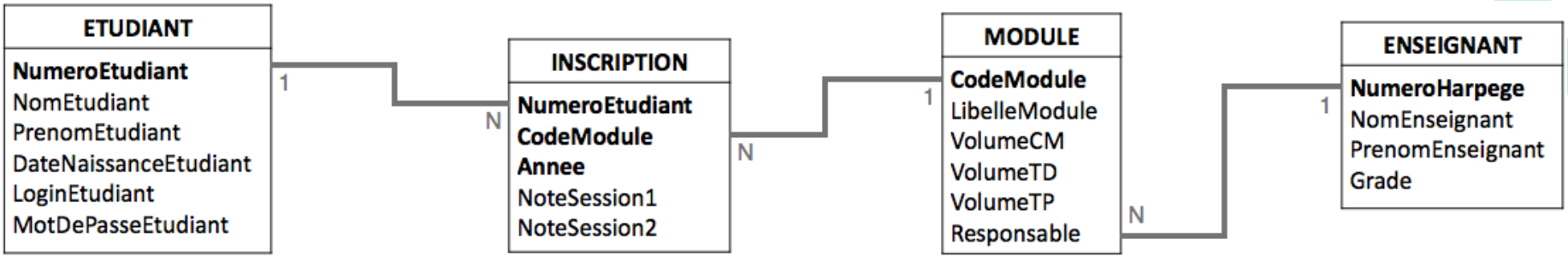
Les résultats d'une sous-requête : _____

_____ 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')
```

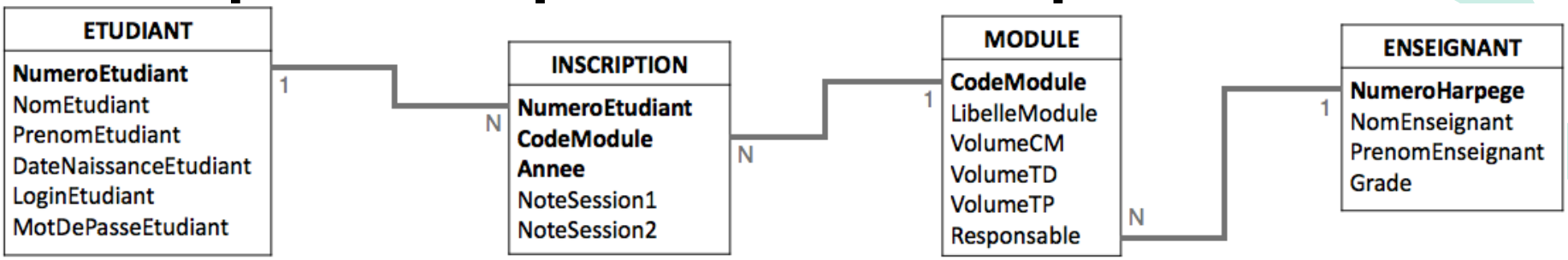
- Code avec sous-requête : plus lisible, plus simple que celui sans.

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

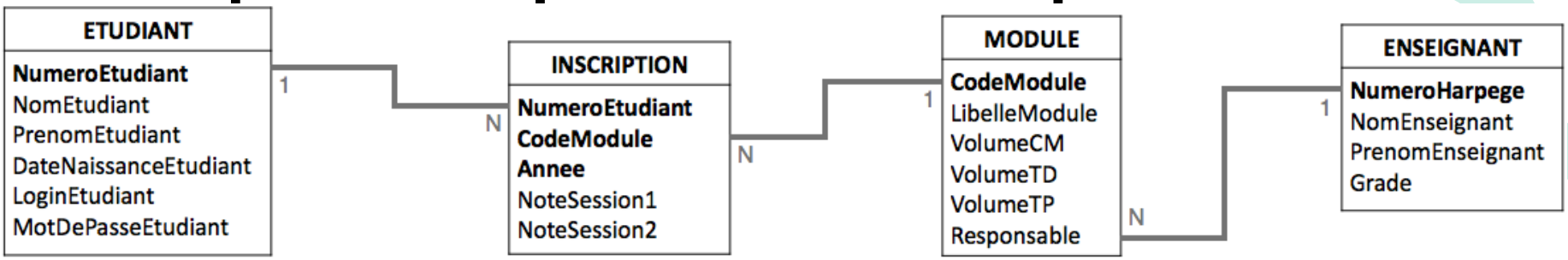
Module

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

1. Requêtes imbriquées :

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

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

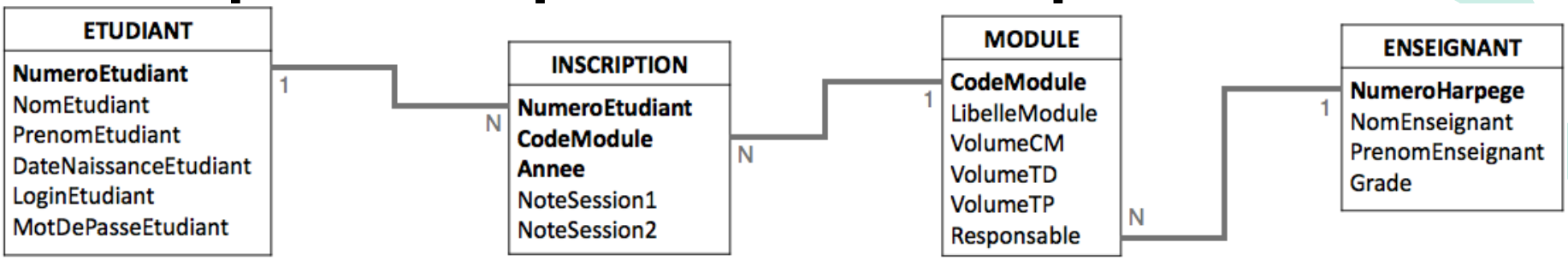
1. Requêtes imbriquées :

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

2. Sous-requête :

```
SELECT Responsable FROM Module WHERE CodeModule = 'BD_L1'
```
$\rightsquigarrow$ 7914

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

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')
```

2. Sous-requête :

```
SELECT Responsable FROM Module WHERE CodeModule = 'BD_L1'
```
$\rightsquigarrow$ 7914

3. Requête principale :

```
SELECT CodeModule FROM Module WHERE Responsable = 7914
```

Sous-requête indépendante : type du résultat

Sous-requête indépendante retournant un seul *n*-uplet

- Emploi des opérateurs de comparaison entre 2 *n*-uplets : $=, <>, <, <= \ldots$
- 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)
```

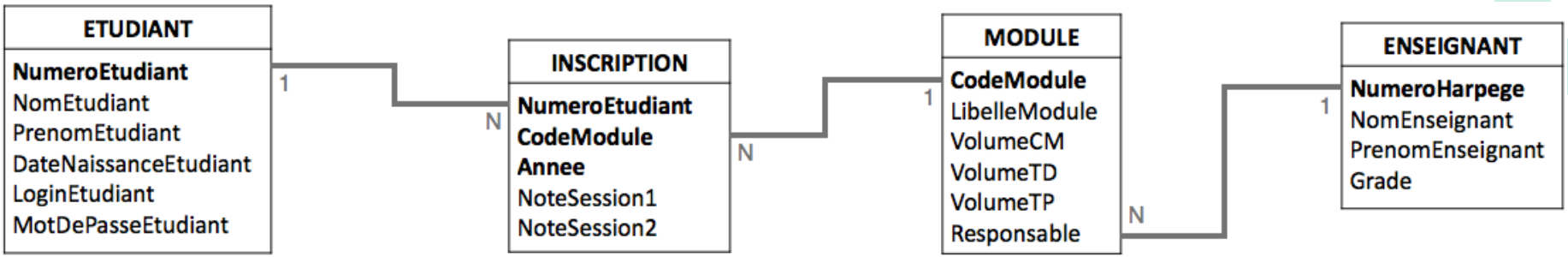
Sous-requête indép. retournant un ensemble de *n*-uplets

- 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 : $=, <>, <, <= \ldots$
  - suivi de ANY ou ALL

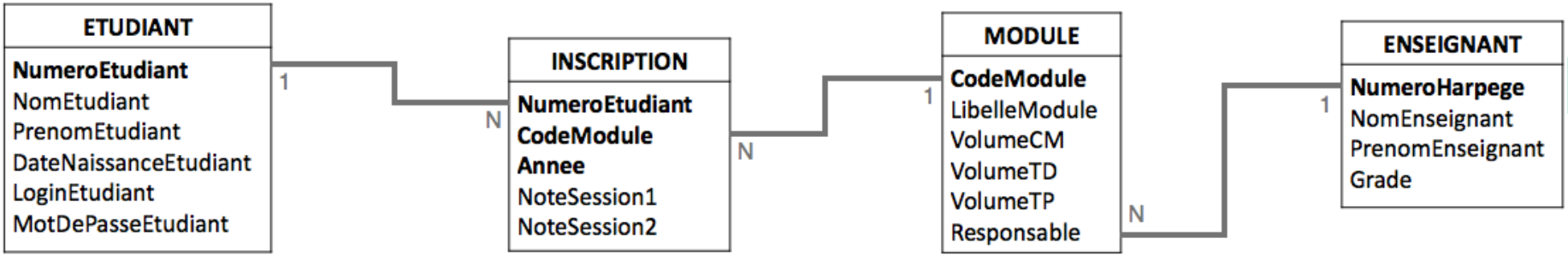
- op ANY (SELECT ... FROM ... WHERE ...) : vraie si la condition utilisant op est vraie pour_____

  - Rem. : comparaison fausse 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

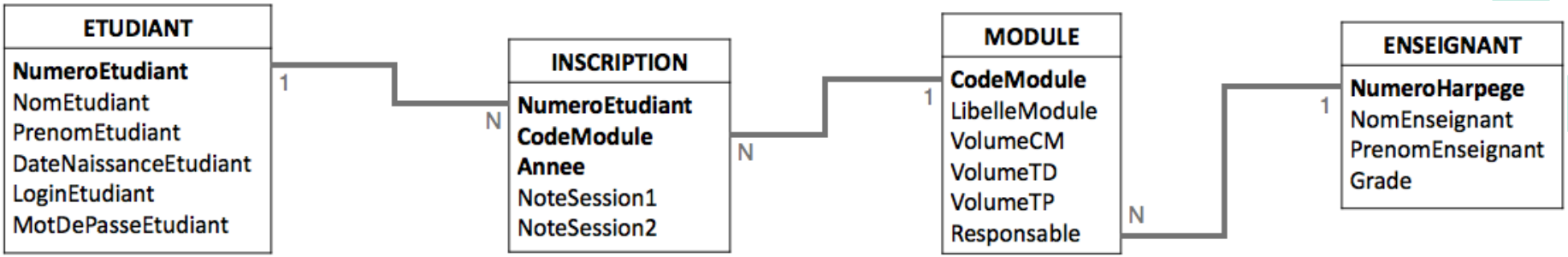
```
SELECT NumeroHarpege FROM ENSEIGNANT, MODULE
WHERE NumeroHarpege = Responsable GROUP BY NumeroHarpege
HAVING COUNT(CodeModule) >= ALL (SELECT COUNT(CodeModule) FROM MODULE
                                 GROUP BY Responsable)
```

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 :

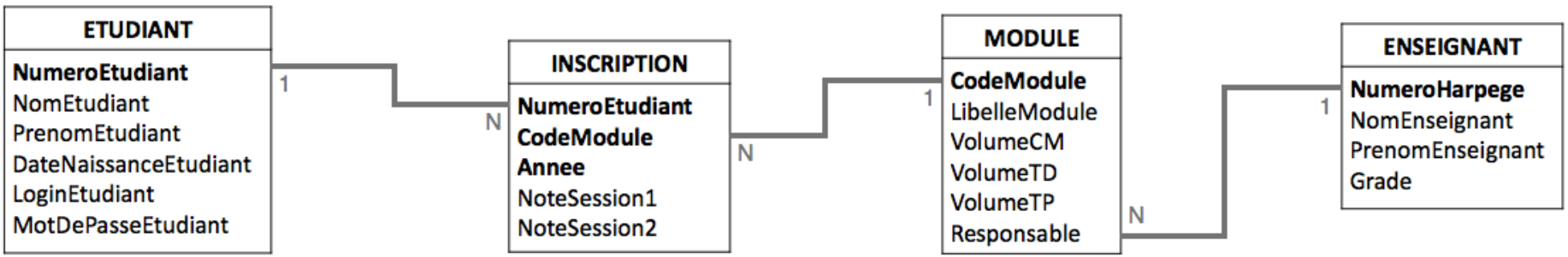
```
SELECT DISTINCT NumeroEtudiant FROM INSCRIPTION AS INSCR
WHERE NoteSession1 > (SELECT AVG(NoteSession1) FROM INSCRIPTION
                      WHERE CodeModule = INSCR.CodeModule)
```

- Sous-requête : _____
  _____

Sous-requêtes synchronisées avec EXISTS

Définition

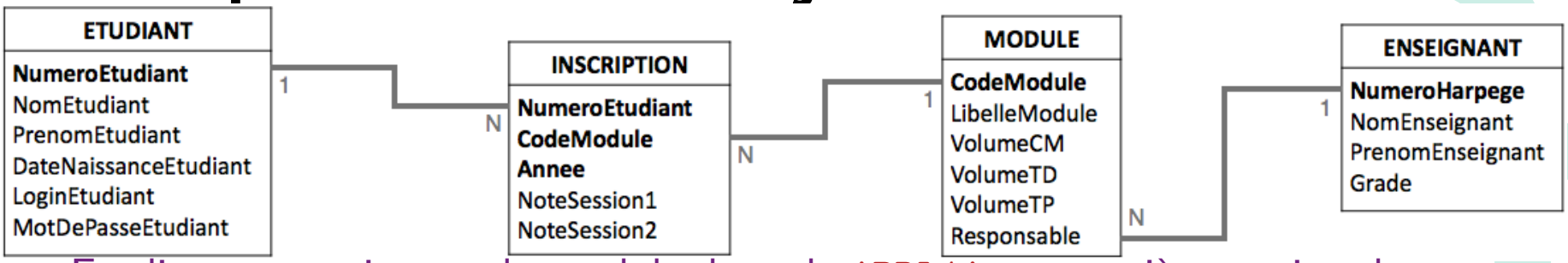
- Syntaxe : `SELECT ... WHERE [NOT] EXISTS (Sous-requête synchronisée)`
- `EXISTS` : vraie si la sous-requête synchronisée renvoie au moins un résultat



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.Annee = 2022
                  AND INSCRIPTION.NumeroEtudiant = ETUDIANT.NumeroEtudiant)
```

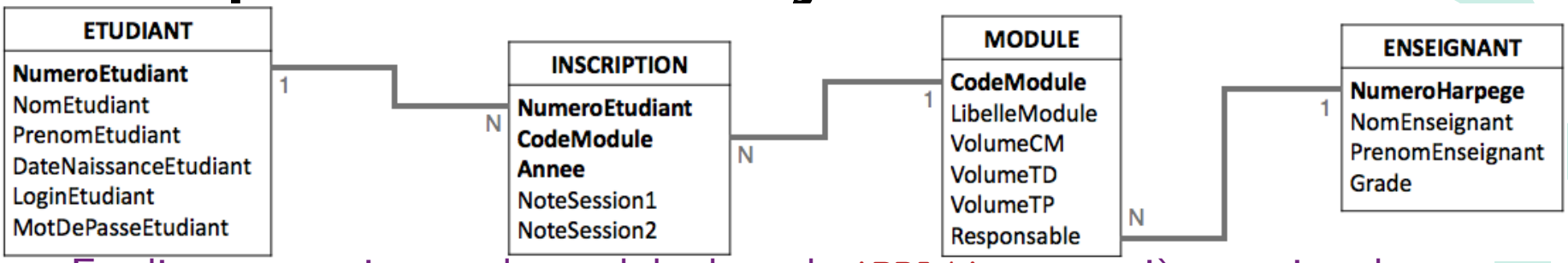
Sous requêtes : à vous de jouer



Etudiant·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

Sous requêtes : à vous de jouer

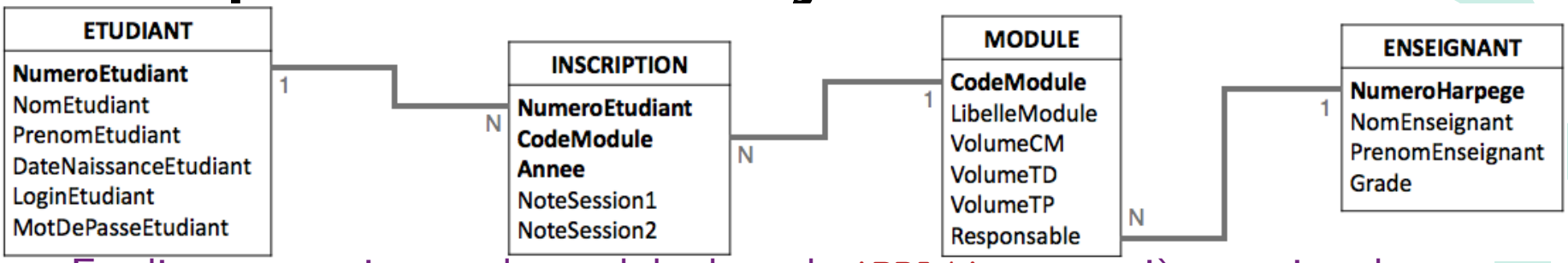


Etudiant·e·s major·e·s du module de code `'BDL1'` en première session de tous les temps

- _____

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

Sous requêtes : à vous de jouer



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

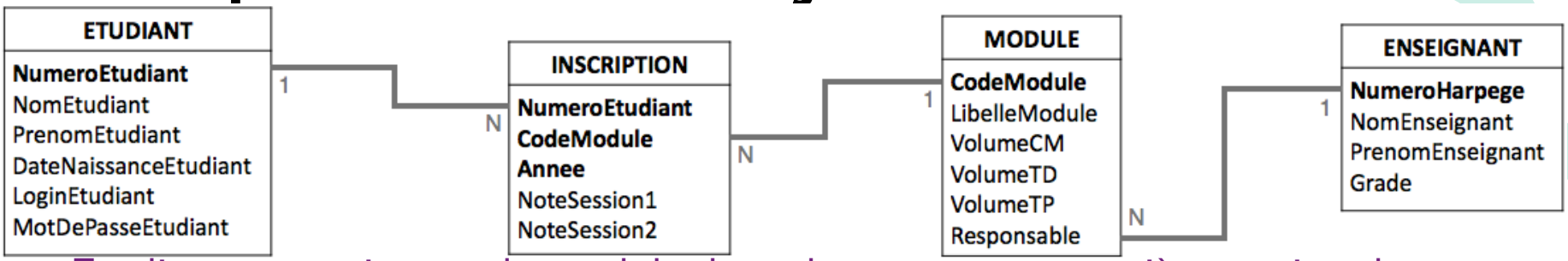
- _____
- _____

_____

_____

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

Sous requêtes : à vous de jouer



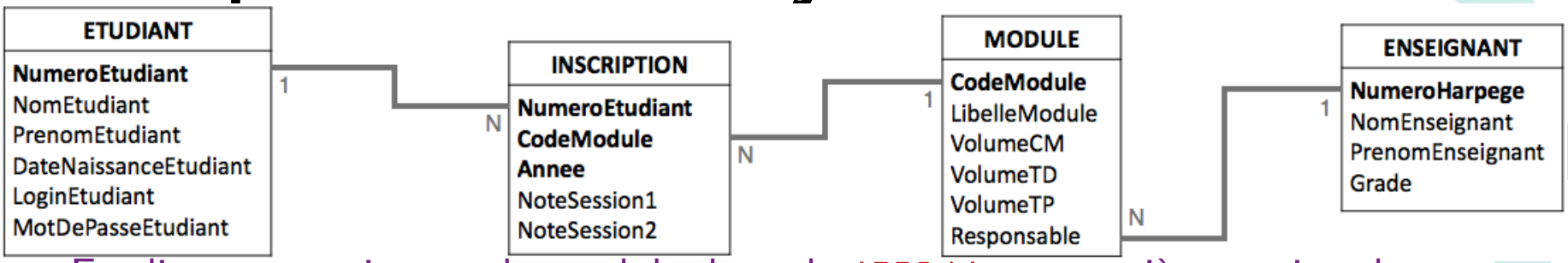
Etudiant·e·s major·e·s du module de code `'BDL1'` en première session de tous les temps

- _____
- _____
  _____
  _____

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

- _____

Sous requêtes : à vous de jouer



Etudiant·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 : _____, _____

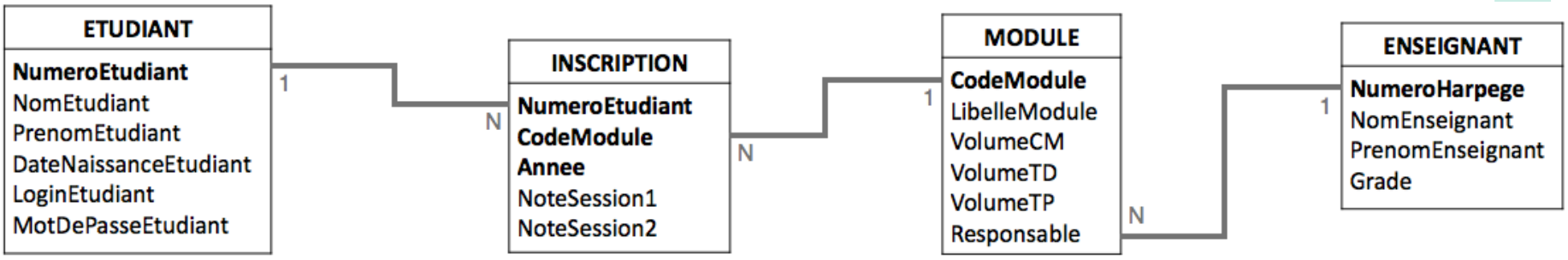
Syntaxe

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

Remarques

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

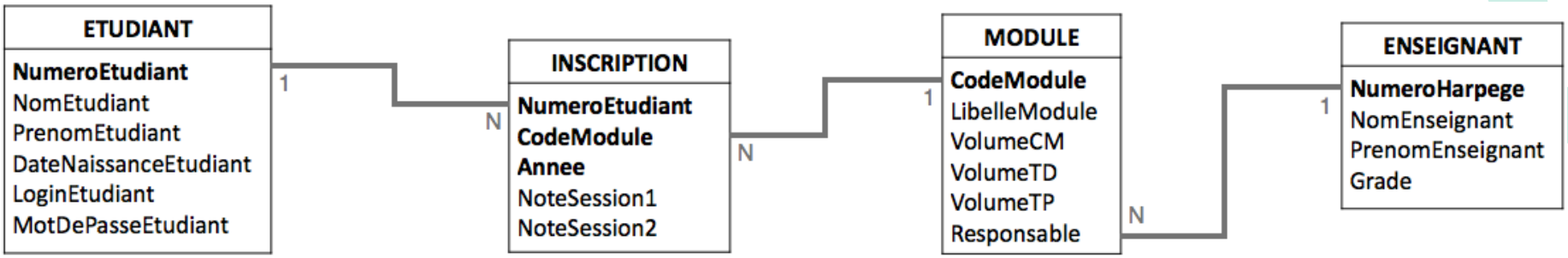
Exemple avec UNION



Noms, prénoms des enseignant·e·s 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

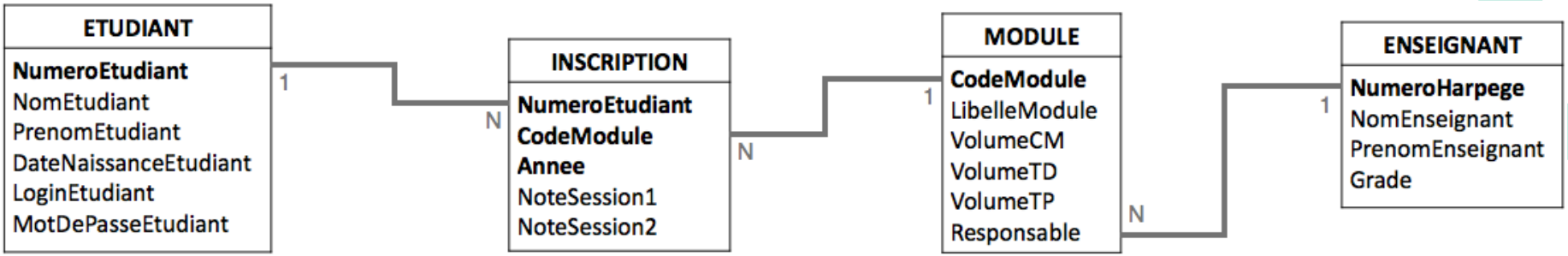
```
SELECT NomEnseignant, PrenomEnseignant FROM ENSEIGNANT WHERE Grade='MCF'
INTERSECT
SELECT NomEnseignant, PrenomEnseignant FROM ENSEIGNANT INNER JOIN MODULE
ON NumeroHarpage = Responsable WHERE VolumeTD = 0
```

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 Gra
EXCEPT
SELECT NumeroHarpege, NomEnseignant, PrenomEnseignant FROM ENSEIGNANT INNER JOI
ON NumeroHarpage = 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 NumeroHarpage = 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 $R_1 \subset A_1 \times \cdots \times A_m \times B_1 \times \cdots \times B_n$ par $R_2 \subset B_1 \times \cdots \times B_n$

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

$$\begin{array}{l|l} R_1 & R_2 \\ \hline \ldots & Q \\ R & \end{array} \quad \text{avec } R_1 = Q \times R_2 \ \texttt{union}\ R$$

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

INSCRIPTION =

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

MOD1

CodeModule*
BD_L1
PROG_L1
TEST_L3

INSCRIPTION DIV MOD1

NoEtudiant*
32911
34812

Traduction ensembliste

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

Division : 2 sous-req. NOT EXISTS imbriquées

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

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

$$\Leftrightarrow$$

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

$$\Leftrightarrow$$

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

Traduction SQL $\rightsquigarrow$ formule à particulariser

```
SELECT A1,...,Am FROM R1          WHERE
   NOT EXISTS (SELECT B1,...,Bn FROM R2          WHERE
      NOT EXISTS (SELECT A1,...,Am,B1,...,Bn FROM R1 AS R1P
                  WHERE R1.A1 = R1P.A1 AND ... R1.Am = R1P.Am AND
                        R2.B1 = R1P.B1 AND ... R2.Bn = R1P.Bn))
```

Division : 2 sous-req. NOT EXISTS imbriquées

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

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

$$\Leftrightarrow$$

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

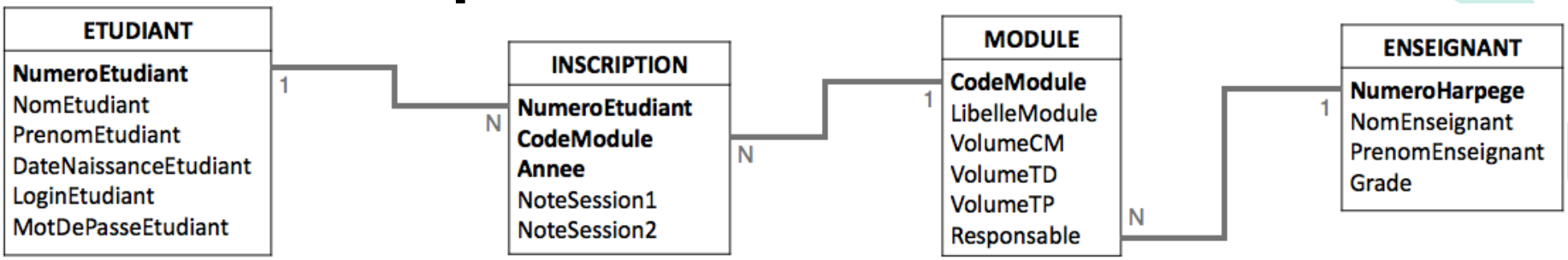
$$\Leftrightarrow$$

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

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



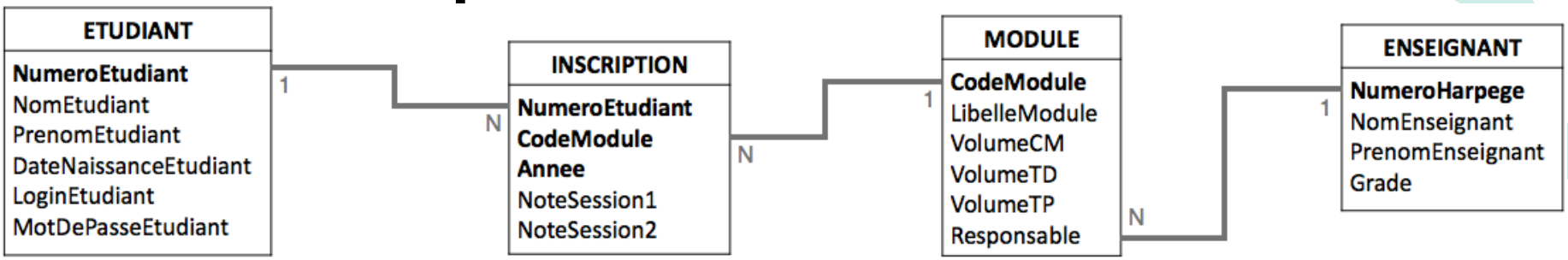
Formule à particulariser

```
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 _____

Division : exemple



Formule à particulariser

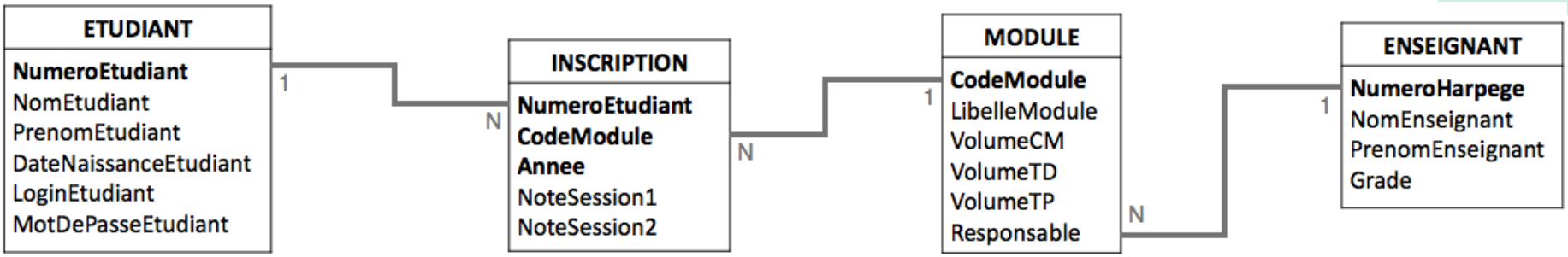
```
SELECT A1,...,Am FROM R1 AS T1  WHERE
   NOT EXISTS (SELECT B1,...,Bn FROM R2 AS T2  WHERE
      NOT EXISTS (SELECT A1,...,Am,B1,...,Bn FROM R1 AS T1P
                  WHERE T1.A1 = T1P.A1 AND ... T1.Am = T1P.Am AND
                        T2.B1 = T1P.B1 AND ... T2.Bn = T1P.Bn))
```

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

- R1 _____
- R2 _____

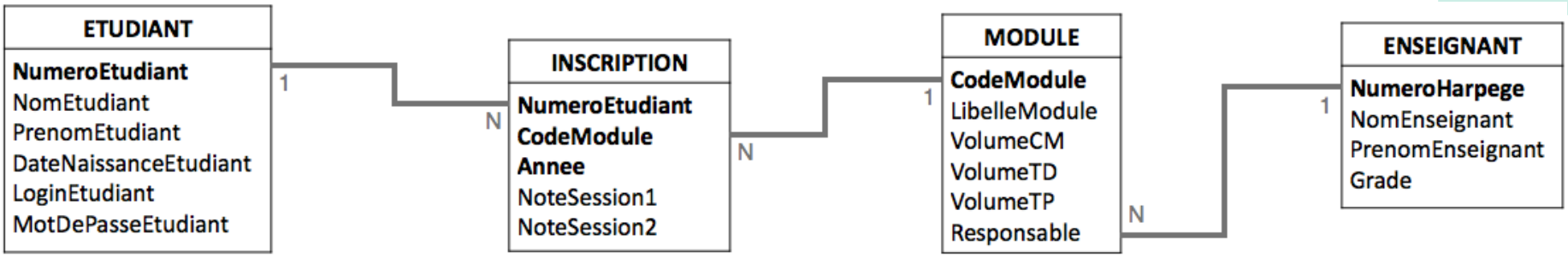
```
SELECT _____ FROM _____ AS T1 WHERE
 NOT EXISTS (SELECT _____ FROM MODULE AS T2 WHERE
 NOT EXISTS (SELECT _____, _____ FROM _____ AS T1P
WHERE T1._____ = T1P._____ AND T2._____
= T1P._____))
```

Division : à vous de jouer



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

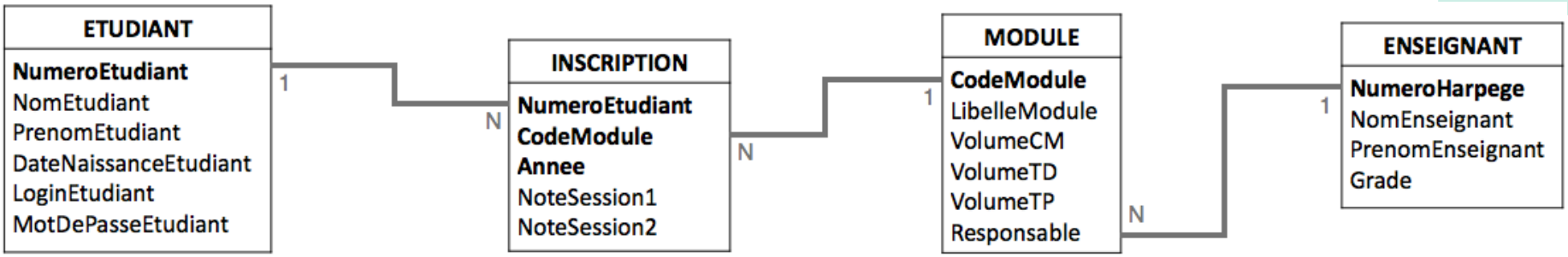
Division : à vous de jouer



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

- _____
- _____

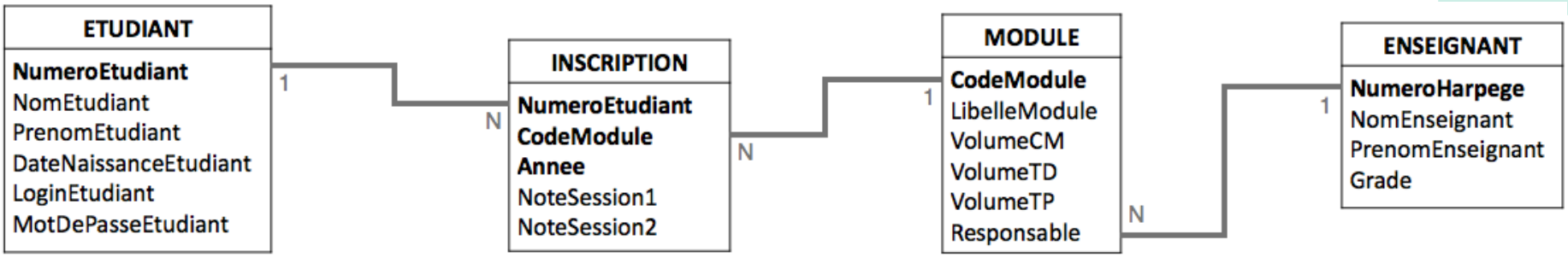
Division : à vous de jouer



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

```
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))
```

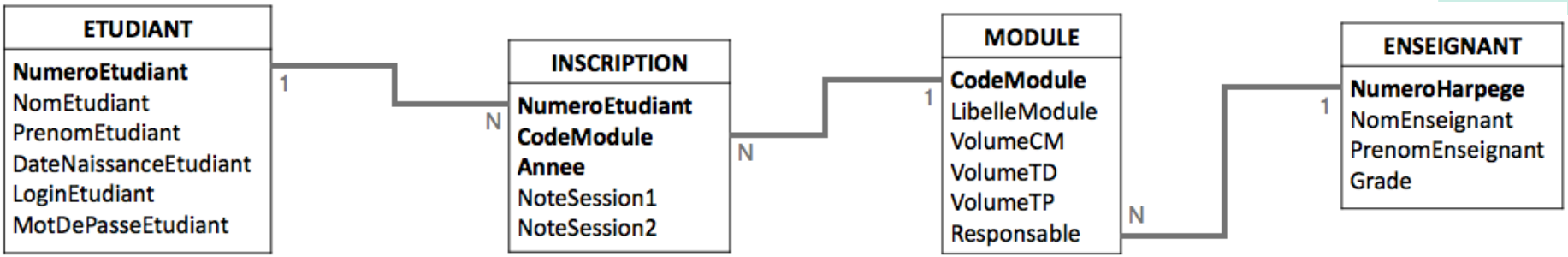
Division : à vous de jouer



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

```
SELECT NumeroEtudiant FROM INSCRIPTION AS T1  WHERE
   NOT EXISTS (SELECT B1,...,Bn FROM R2 AS T2  WHERE
      NOT EXISTS (SELECT NumeroEtudiant,B1,...,Bn FROM INSCRIPTION AS T1P
                  WHERE T1.NumeroEtudiant = T1P.NumeroEtudiant AND
                        T2.B1 = T1P.B1 AND ... T2.Bn = T1P.Bn))
```

Division : à vous de jouer



Numéros des étudiant·e·s inscrit·e·s 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

Fonctions et expressions

Rappel sur les fonctions de groupe

- Applicables _____

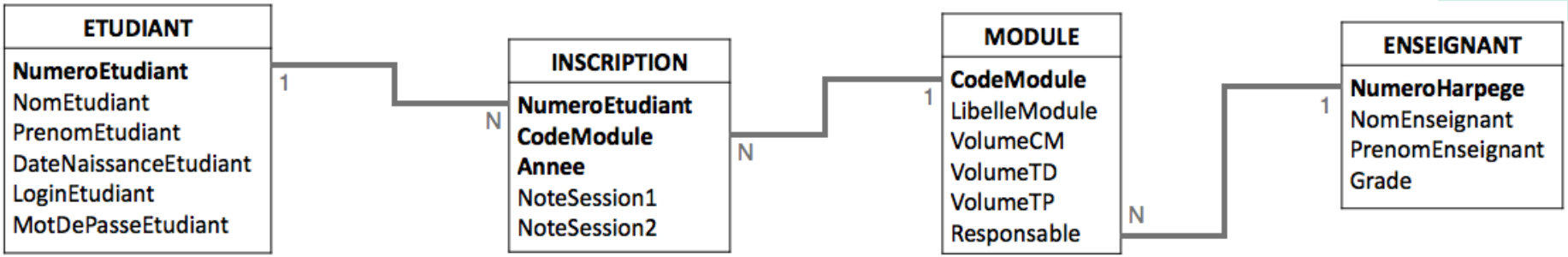
Expressions et fonctions

- 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

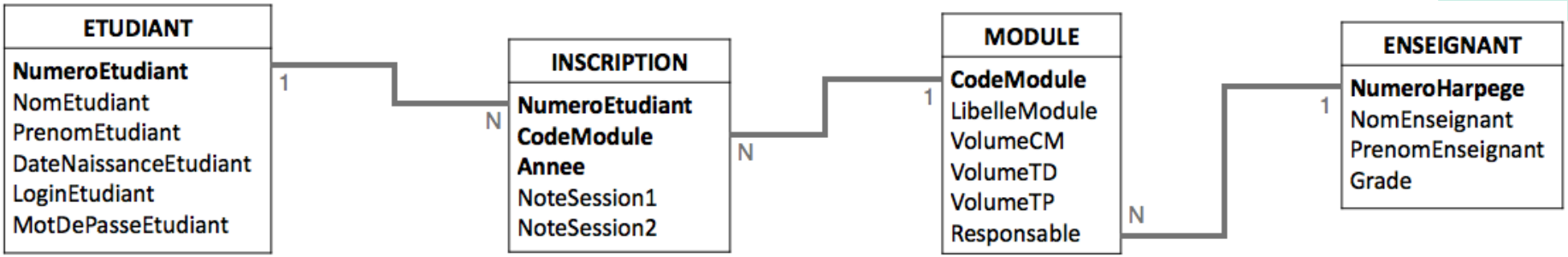
MODULE

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

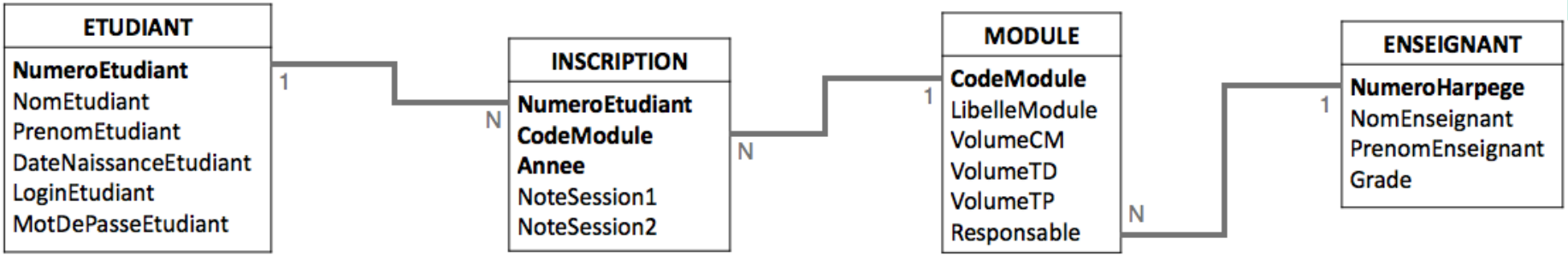
MODULE

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

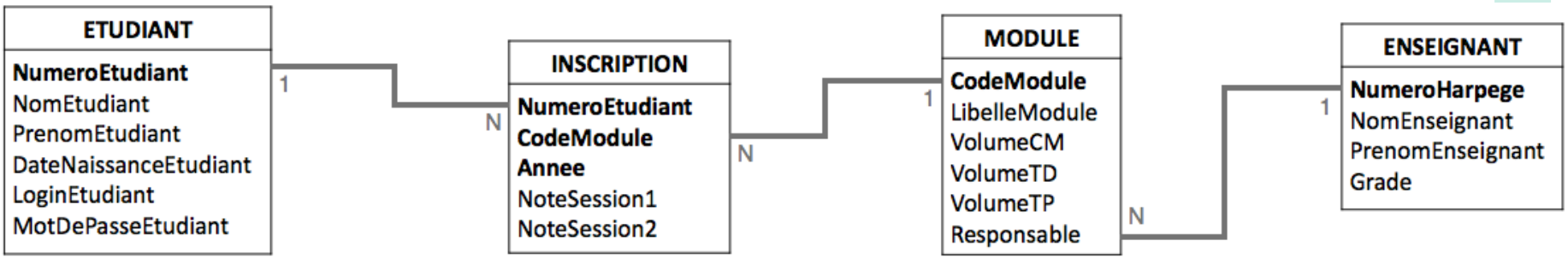
MODULE

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)`,...

Plan

Introduction

Data Query Language

Data Manipulation Language
- Ajout de données
- Modification de données
- Suppression de données

Data Definition Language

Data Control Language

Plan

Introduction

Data Query Language

Data Manipulation Language
- Ajout de données
- Modification de données
- Suppression de données

Data Definition Language

Data Control Language

Ajout de données avec VALUES

Syntaxe

```
INSERT INTO Table [(Att1, Att2,...)] VALUES (x1, x2,...), (y1, y2,...),...
```

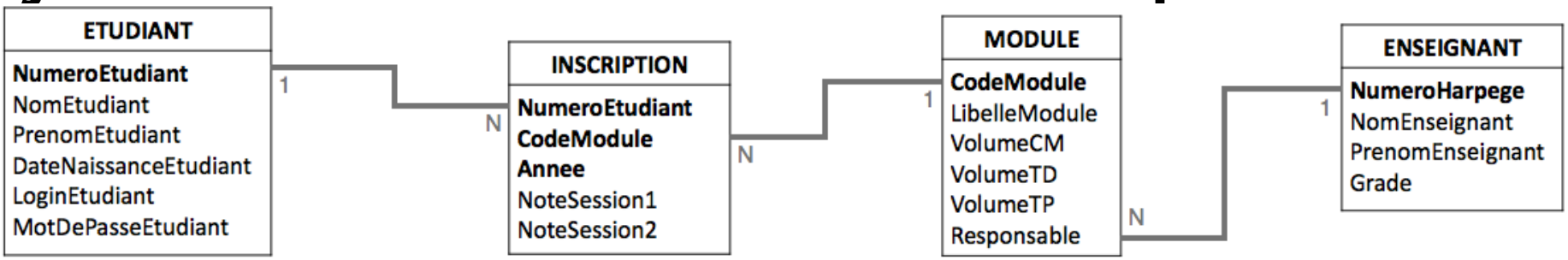
Liste facultative d'attributs `(Att1, Att2, ...)`

- Restreindre les attributs à renseigner dans le nouvel enregistrement
- Attributs non renseignés : _____
- Si liste absente : _____

Liste obligatoire de valeurs `(x1, x2,...), (y1, y2, ...), ...`

- Ordre des valeurs : _____ celui des attributs (`xn` $\leftrightarrow$ `Attn` par ex.).
- Chaque valeur `xn` : doit _____ au domaine de `Attn`
- Insertion multiples : n-uplets _____

Ajout de données avec VALUES : exemple



Exemple d'insertions d'enseignants

ENSEIGNANT

NumeroHarpege	NomEnseignant	PrenomEnseignant	Grade
7358	Féléa	Violeta	MCF
7914	Dadeau	Frédéric	MCF

```
INSERT INTO ENSEIGNANT (NumeroHarpege, NomEnseignant, PrenomEnseignant)
   VALUES (32598, 'Paquette', 'Guillaume')

INSERT INTO ENSEIGNANT VALUES (1797, 'Damy', 'Sylvie', 'MCF')
```

ENSEIGNANT

NoHarpege*	Nom	Prenom	Grade
7358	Féléa	Violeta	MCF
7914	Dadeau	Frédéric	MCF
32598	Paquette	Guillaume	NULL
1797	Damy	Sylvie	MCF

Ajout de données issues de SELECT : définition

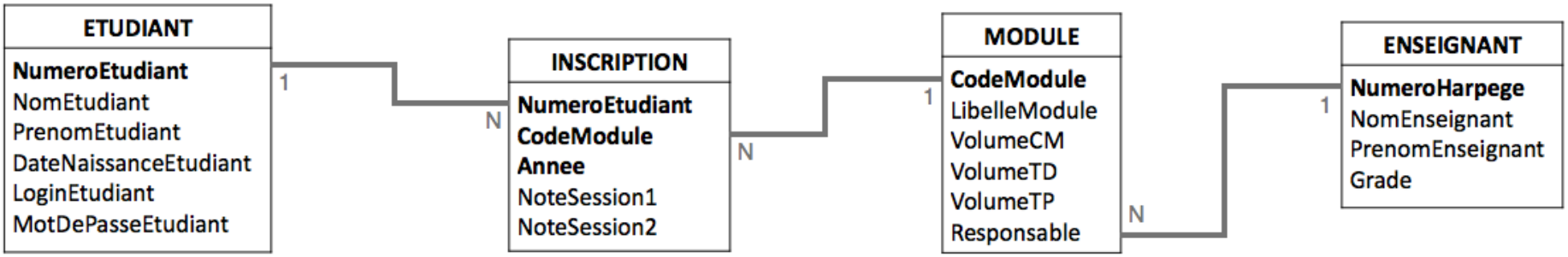
Syntaxe

```
INSERT INTO Table [(Att1, Att2, ...)]  SELECT Att1, Att2, ...
```

Remarques

- *n*-uplets retournés par SELECT : _____
- Instruction SELECT : ____ instruction ORDER BY.

Ajout de données issues de SELECT : exemple



Réinscription en 2023 des étudiant·e·s dans les modules non validés de 2022

```
INSERT INTO INSCRIPTION (NumeroEtudiant, CodeModule, Annee)
   SELECT NumeroEtudiant, CodeModule, Annee + 1 FROM INSCRIPTION
   WHERE Annee = 2022 AND
   ((NoteSession1 < 10 AND NoteSession2 IS NULL) OR (NoteSession2 < 10))
```

Plan

Introduction

Data Query Language

Data Manipulation Language
- Ajout de données
- Modification de données
- Suppression de données

Data Definition Language

Data Control Language

Modification de données avec UPDATE : définition

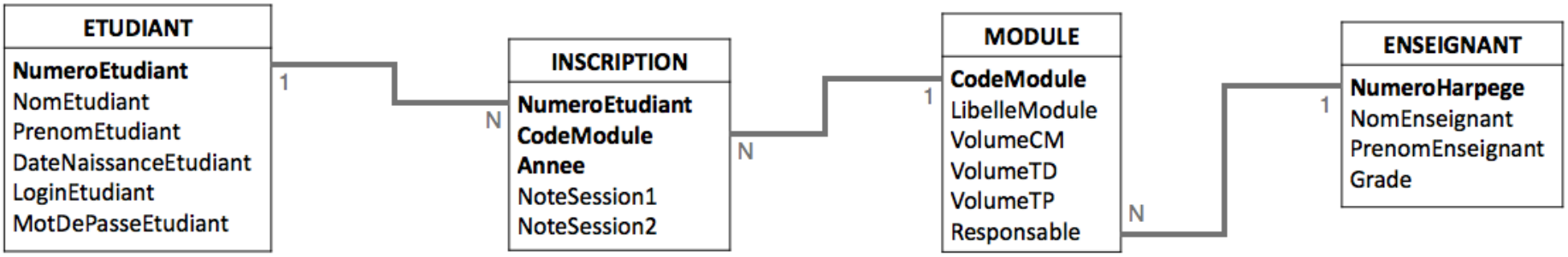
Syntaxe

```
UPDATE Table  SET Att1 = Val1, Att2 = Val2, ... WHERE Condition
```

Remarque

- Mise à jour des n-uplets de `Table` _____
- Valeurs modifiées : _____

Modification de données avec UPDATE : exemple



'@L1Sc1enc3 !' : nouveau mot passe de l'étudiant 23794

- Code :

  _____

  _____

- Remarque : _____ *n*-uplet modifié

Responsabilités assumées par 7914 : assumées par 32598

- Code :

  _____

- Remarque : _____*n*-uplets modifiés

Plan

Introduction

Data Query Language

Data Manipulation Language
- Ajout de données
- Modification de données
- Suppression de données

Data Definition Language

Data Control Language

Suppression avec DELETE : définition

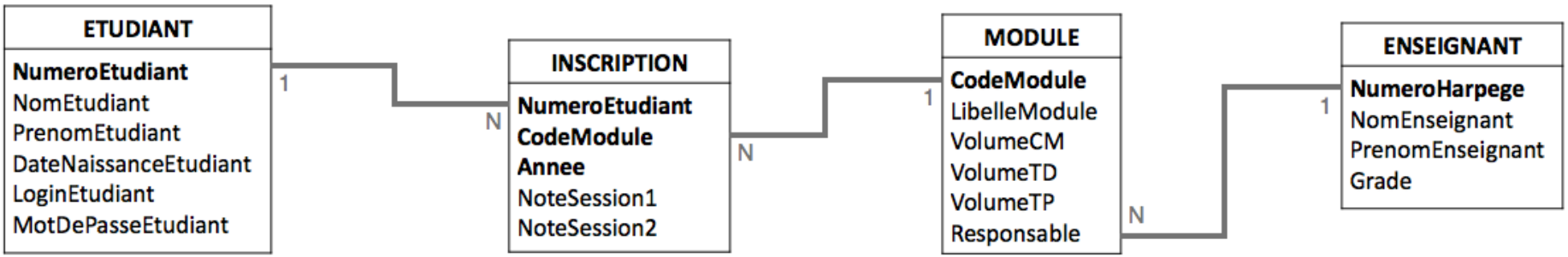
Syntaxe

```
DELETE FROM Table WHERE Condition
```

Remarque

- Tous les *n*-uplets de `Table` qui satisfont la condition du `WHERE` : supprimés

Suppression de données avec DELETE : exemples

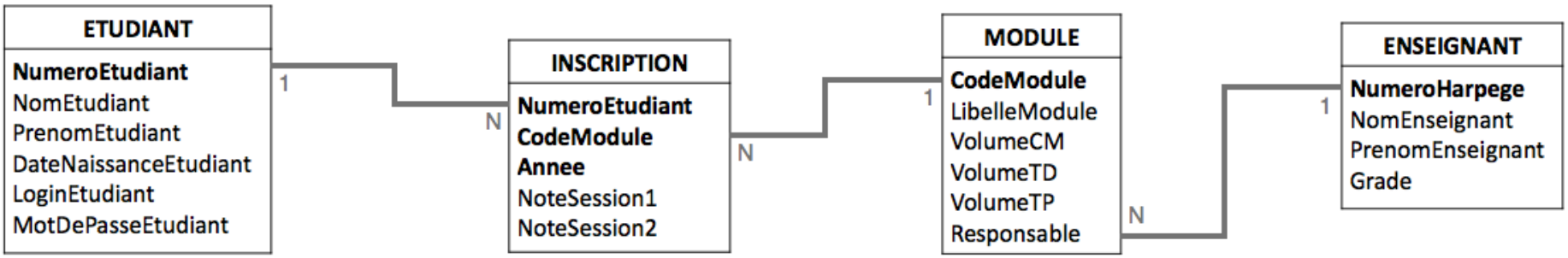


Suppression de l'enseignant Jean-François Couchot

Suppression des étudiant·e·s inscrit·e·s nulle part

Suppression des modules sans inscrit·e·s depuis 2013

Suppression de données avec DELETE : exemples

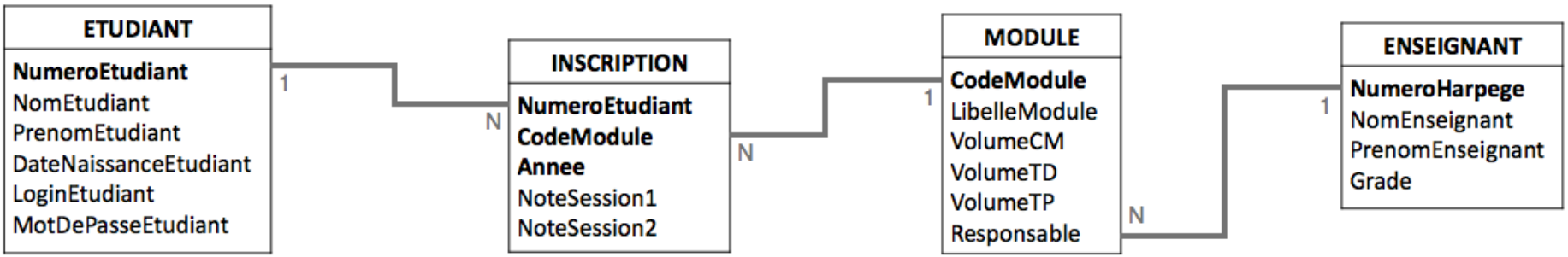


Suppression de l'enseignant Jean-François Couchot

Suppression des étudiant·e·s inscrit·e·s nulle part

Suppression des modules sans inscrit·e·s depuis 2013

Suppression de données avec DELETE : exemples



Suppression de l'enseignant Jean-François Couchot

Suppression des étudiant·e·s inscrit·e·s nulle part

Suppression des modules sans inscrit·e·s depuis 2013

INSERT, UPDATE, DELETE : intégrité

Requêtes INSERT, UPDATE, DELETE violant l'intégrité référentielle

- INSERT d'une valeur déjà existante pour une clé primaire
- UPDATE de la valeur d'un attribut clé primaire référencée dans une table liée, avec une politique de mise à jour RESTRICT
- DELETE d'un enregistrement dont la clé primaire est référencée dans une table liée, avec une politique de suppression RESTRICT

vont toujours produire des erreurs renvoyées par le SGBD !

Plan

Introduction

Data Query Language

Data Manipulation Language

Data Definition Language
- Tables
- Index

Data Control Language

Plan

Introduction

Data Query Language

Data Manipulation Language

Data Definition Language
- Tables
- Index

Data Control Language

Création d'une table : CREATE TABLE

Description générale

- des attributs de la table (leur nom, leur type)
- des contraintes d'intégrité de cette table

Syntaxe basique

```
CREATE TABLE NOM_TABLE
 (NomAttribut1 Type1,
  NomAttribut2 Type2,
  ...)
```

- Type1, Type2 : types dépendant du SGBD (MySQL $\neq$ ORACLE...)
- Types communs : INTEGER, CHAR(length), DATE, ENUM(v1, v2, ...)

Exemple : création de la table INTERVENANT listant les enseignant·e·s intervenant dans un module

```
CREATE TABLE INTERVENANT
   (CodeModule CHAR(5),
    Enseignant INTEGER)
```

Créat° & rempliss. : CREATE TABLE...AS SELECT

Syntaxe de création et remplissage initial d'une table

```
CREATE TABLE NOM_TABLE
  (NomAttribut1 Type1,
   NomAttribut2 Type2,
   ...)
AS SELECT ...
```

Remarques sur cette syntaxe

- Noms des attributs de la table créée : ceux des attributs du SELECT

Créat° d'INTERVENANT et initialisat° avec les resp. de modules

```
CREATE TABLE INTERVENANT
  (CodeModule CHAR(5),
   Enseignant INTEGER)
AS SELECT CodeModule, Responsable FROM MODULE
```

Ajout de contraintes d'intégrité sur les attributs

Syntaxe

```
CREATE TABLE NOMTABLE
  (Att1 Type1 Contrainte1 ... ContrainteN,
   ...)
```

Différents types de contraintes sur les attributs

- `Att Type DEFAULT valParDef` : `Att` _____ dans `INSERT`
- `Att Type NOT NULL` : `Att` _____ `NULL`
- `Att ... UNIQUE` : deux occurr. d'une m̂ valeur _____
- `Att ... CHECK (condition)` : condition _____ de `Att`

Exemple de création de la table ETUDIANT

```
CREATE TABLE ETUDIANT
  (NumeroEtudiant INTEGER PRIMARY KEY,
   NomEtudiant VARCHAR(50) NOT NULL,
   PrenomEtudiant VARCHAR(50) NOT NULL,
   DateNaissanceEtudiant DATE CHECK (DateNaissanceEtudiant<CURRENT_DATE),
   LoginEtudiant CHAR(8) NOT NULL UNIQUE,
   MotDePasseEtudiant CHAR(64) NOT NULL)
```

Contraintes d'intégrité sur les tables

Syntaxe : après la définition des attributs

```
CREATE TABLE NOMTABLE
  (Att1 Type1 CA1,
   ...,
   Attk Typek CAk,
   [CONSTRAINT nom_contrainte_table_1] CONTRAINTE_TABLE_1,
   ...,
   [CONSTRAINT nom_contrainte_table_l] CONTRAINTE_TABLE_l)
```

Remarques

- `CONSTRAINT nom_contrainte_table_i` : _____
- `CONTRAINTE_TABLE` : peuvent concerner les clefs
  - `PRIMARY KEY (Att1, Att2,...)` : définit° de la clé primaire
    - _____ `PRIMARY KEY` par table
    - `NOT NULL` _____ pour chaque attribut de la clef
  - `FOREIGN KEY (Atti) REFERENCES NOMAUTRETABLE(Attj)`
    `[ON UPDATE ...] [ON DELETE ...]`
    - Lien entre `NOMAUTRETABLE.Attj`,_____ et `Atti`, _____
    - `ON UPDATE ON DELETE` : _____ `RESTRICT` ou `CASCADE`

Expl. de créatº de table avec des contraintes

Exemple de création de la table INTERVENANT

```
CREATE TABLE INTERVENANT
  (CodeModule CHAR(5),
   Enseignant INTEGER,
   CONSTRAINT Pk_intervenant PRIMARY KEY (CodeModule, Enseignant),
   CONSTRAINT Fk_module FOREIGN KEY (CodeModule)
                        REFERENCES MODULE(CodeModule)
                        ON DELETE CASCADE ON UPDATE CASCADE,
   CONSTRAINT Fk_enseignant FOREIGN KEY (Enseignant)
                            REFERENCES ENSEIGNANT(NumeroHarpege)
                            ON DELETE CASCADE ON UPDATE CASCADE)
```

Modification de la structure d'une table

3 types de modification de la structure d'une table

- l'ajout d'un attribut : `ADD`
- la modification d'un attribut existant : `MODIFY`
- la suppression d'un attribut : `DROP`

Modification de la structure d'une table

Syntaxe

`ALTER TABLE NOMTABLE instruction : instruction est`

- `ADD (Att1 Type1 [Contraintes], Att2 ...)` : ajout d'attributs (cf. `CREATE TABLE`)
- `MODIFY (Att1 Type1 [Contraintes], Att2 ...)` : modification du type et des contraintes
- `DROP Att1, DROP Att2...` : suppression des attributs

Exemples de modifications

- Ds INTERVENANT : +attribut pr. les heures de (module,enseignant)

```
ALTER TABLE INTERVENANT ADD (NbHeures INTEGER CHECK (NbHeures > 0))
```

- Modification d'ETUDIANT : prise en compte des noms longs (100)

```
ALTER TABLE ETUDIANT MODIFY NomEtudiant char(100) NOT NULL
```

- Ds ETUDIANT : suppression des dates de naissance

```
ALTER TABLE ETUDIANT DROP DateNaissanceEtudiant
```

Suppression d'une table

Syntaxe

```
DROP TABLE NOMTABLE
```

Remarque

- `DROP` d'une table avec un attribut référencé dans une autre : refusé

Exemple : suppression de la table INTERVENANT

```
DROP TABLE INTERVENANT
```

Plan

Introduction

Data Query Language

Data Manipulation Language

Data Definition Language
- Tables
- Index

Data Control Language

Index : motivations

Accès lent à certains enregistrements d'une table

```
SELECT * FROM ETUDIANT WHERE NomEtudiant = ``Smolinski''
```

$\rightsquigarrow$ parcours séquentiel du SGBD de tous les enregistrements jusqu'à `NomEtudiant = 'Smolinski'` ? Non !

Accès rapide à certains enregistrements d'une table

- Une structure stockant les adresses des enregistrements organisée en fonction des noms :
- Par dichotomie sur les noms triés par ordre alphabétique : en $O(\log_2(n))$:

Nb. tuples	Nb. itérations	Gain (%)
5	3	40
10	4	60
50	6	88
100	7	93
500	9	98,2
1 000	10	99
5 000	13	99,74
10 000	14	99,86

Index : définition et réflexions

Définition

Un index : une structure entretenue automatiquement, permettant de localiser facilement des enregistrements dans un fichier selon un (ensemble d') attribut(s).

Conséquences

- Lors d'un `CREATE TABLE` : déclaration des attributs composant l'index
- Si pas d'index déclaré : indexation selon la clef primaire
- A chaque `INSERT`, `UPDATE`, `DELETE` : mise à jour de l'index par le SGBD
- Requêtes `SELECT` : transparentes à l'existence d'un index
- Intéressant de créer un index si :
  - Sélections récurrentes sur un ensemble d'attributs
  - Jointures récurrentes entre deux tables p.r. à un ensemble d'attributs
- Contre-productif de créer un index si :
  - la table contient peu d'enregistrements
  - l'attribut ou l'ensemble d'attributs prennent peu de valeurs différentes

Index : syntaxe et exemple

Syntaxe : après la définition des attributs

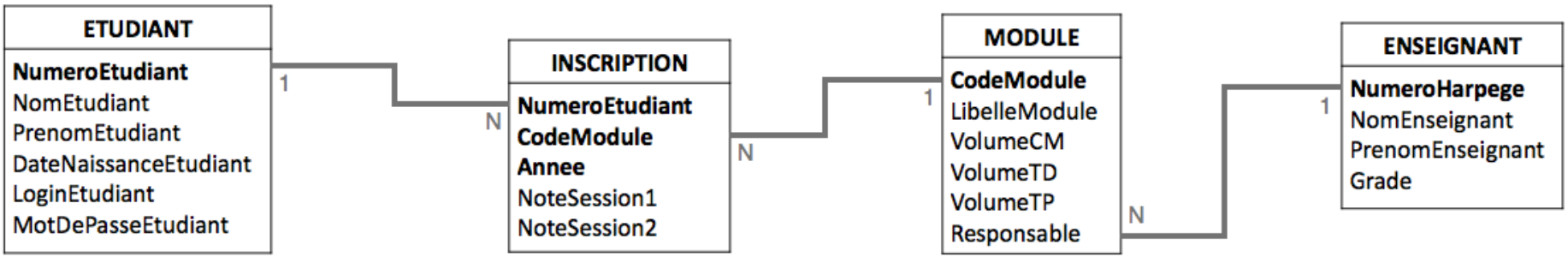
```
CREATE TABLE NOMTABLE
(Att1 Type1 [CA11 ... ],...,
[CONSTRAINT nom_contrainte_table_1] CONTRAINTE_TABLE_1,...,
INDEX [UNIQUE] nom_index (Attj,Attk,...),...)
```

- `Attj`, `Attk` sont les attributs composant l'index
- Option `UNIQUE` : valeurs uniques pour `(Attj,Attk,...)`

Index sur le nom et prénom des étudiants

```
CREATE TABLE ETUDIANT
  (NumeroEtudiant INTEGER NOT NULL,
   NomEtudiant CHAR(30) NOT NULL,
   PrenomEtudiant CHAR(50) NOT NULL,
   DateNaissanceEtudiant DATE CHECK (DateNaissanceEtudiant<CURRENT_DATE),
   LoginEtudiant CHAR(8) NOT NULL UNIQUE,
   MotDePasseEtudiant CHAR(64) NOT NULL DEFAULT
   SHA2(CONCAT(LEFT(NomEtudiant,3),
               LEFT(PrenomEtudiant,3),DAY(DateNaissanceEtudiant)),512),
   CONSTRAINT Pk_no_etudiant PRIMARY KEY (NumeroEtudiant),
   INDEX idx_nom_prenom (NomEtudiant,PrenomEtudiant))
```

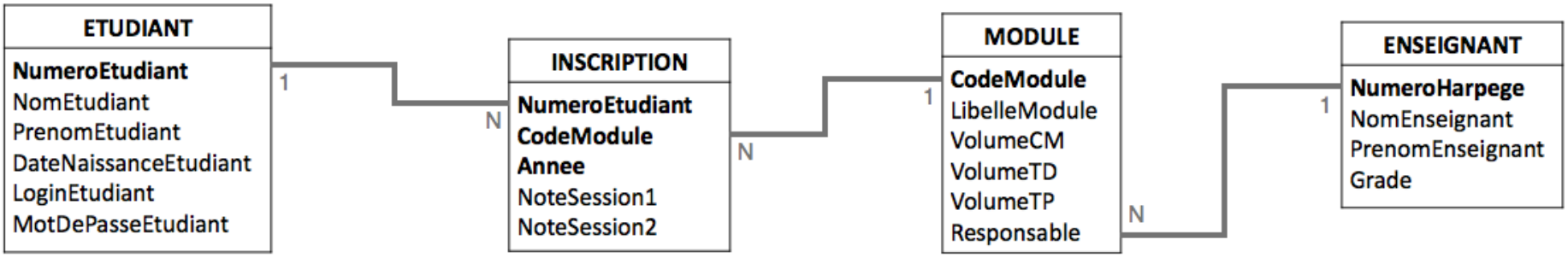
Création : organisation



Préliminaires

- Création ordonnée de : _____
- Index : _____,
  _____,
  _____
- Politique d'intégrité référentielle : _____

Création : ENSEIGNANT et MODULE



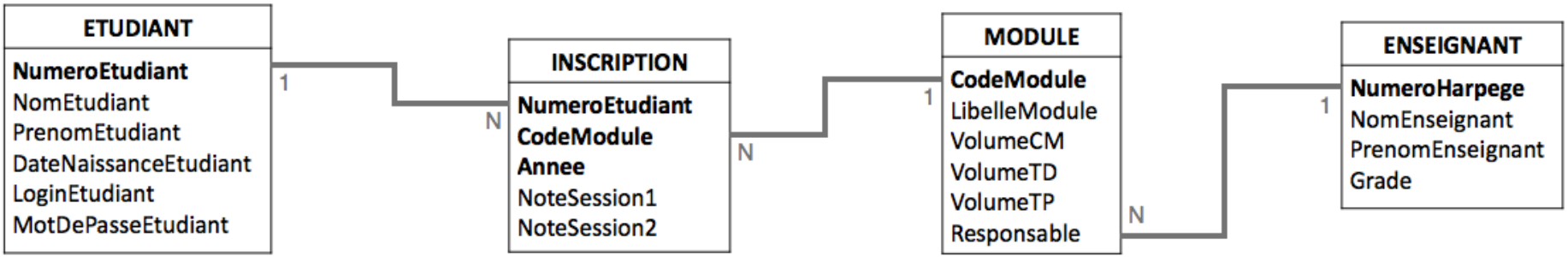
ENSEIGNANT : attributs, clef, index

```
CREATE TABLE ENSEIGNANT
  (NumeroHarpege INTEGER NOT NULL,
   NomEnseignant CHAR(30) NOT NULL,
   PrenomEnseignant CHAR(50) NOT NULL,
   Grade ENUM('ext', 'MCF', 'PU', 'PRAG', 'PRCE', 'MCFPH','PUPH'),
   CONSTRAINT Pk_no_harpege PRIMARY KEY (NumeroHarpege),
   INDEX idx_nom_prenom (NomEnseignant,PrenomEnseignant))
```

MODULE : attributs (avec contraintes), clefs (avec politique), index

```
CREATE TABLE MODULE
  (CodeModule CHAR(10) NOT NULL,
   LibelleModule CHAR(30) NOT NULL,
   VolumeCM INTEGER DEFAULT 0 CHECK (VolumeCM < 100),
   VolumeTD INTEGER DEFAULT 0 CHECK (VolumeTD < 100),
   VolumeTP INTEGER DEFAULT 0 CHECK (VolumeTP < 100),
   Responsable INTEGER NOT NULL,
   CONSTRAINT Pk_code_module PRIMARY KEY (CodeModule),
   CONSTRAINT Fk_resp_no_harpege FOREIGN KEY (Responsable)
                            REFERENCES ENSEIGNANT(NumeroHarpege)
                            ON DELETE CASCADE ON UPDATE CASCADE,
   INDEX idx_libelle (LibelleModule))
```

Création : INSCRIPTION



INSCRIPTION : attributs, clefs (avec politique)

```
CREATE TABLE INSCRIPTION
(NumeroEtudiant INTEGER NOT NULL,
  CodeModule CHAR(10) NOT NULL,
  Annee INTEGER NOT NULL,
  NoteSession1 INTEGER DEFAULT NULL,
  NoteSession2 INTEGER DEFAULT NULL,
  CONSTRAINT Pk PRIMARY KEY (NumeroEtudiant,CodeModule,Annee),
  CONSTRAINT Fk_no_etudiant FOREIGN KEY (NumeroEtudiant)
                            REFERENCES ETUDIANT(NumeroEtudiant)
                            ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT Fk_code_module FOREIGN KEY (CodeModule)
                            REFERENCES MODULE(CodeModule)
                            ON DELETE CASCADE ON UPDATE CASCADE)
```

Plan

Introduction

Data Query Language

Data Manipulation Language

Data Definition Language

Data Control Language
- Utilisateurs
- Attribuer et retirer des droits

Data Control Language

Bilan sur ce qui est (ou devrait être) su

- Création/modification de la structure de données : `CREATE TABLE`, `INDEX`, `DROP TABLE`...
- Manipulation des données : `SELECT`, `UPDATE`, ...

Objectif : contrôler les accès à une base de données

- Définir des utilisateurs
- Leur attribuer/retirer des droits

Plan

Introduction

Data Query Language

Data Manipulation Language

Data Definition Language

Data Control Language
- Utilisateurs
- Attribuer et retirer des droits

Utilisateurs : motivations

Définir des utilisateurs en SQL

- pour autoriser/interdire l'accès à certaines bases
- pour autoriser/interdire certaines opérations sur une base précise
- à partir d'un super-utilisateur `root` avec tous les droits : créer d'autres utilisateurs et leur affecter des droits

Exemple : nécessité de contrôler les utilisateurs

- En TP : accès à la base commune du `CABINETMEDICAL` à partir d'un utilisateur commun
  - autorisé à sélectionner, ajouter, modifier des données
  - non autorisé à modifier la structure
- En L2 Info dans le module "Langage du Web" : accès individuel (un étudiant $\equiv$ un utilisateur MySQL) à une base de données MySQL commune avec droits/interdits comme ci-dessus

Utilisateurs : création, modifications

Syntaxe de création

```
CREATE USER 'Nom' IDENTIFIED BY 'MotDePasse';...
```

Exemple : création de l'utilisateur *SecretaireScol* avec mdp 'scol@ufrST2122 !'

```
CREATE USER 'SecretaireScol' IDENTIFIED BY 'scol@ufrST2122!'
SELECT User, Host, Password FROM mysql.user
  WHERE mysql.user.User='SecretaireScol'
```

←T→	User	Host	Password
Edit Inline Edit Copy Delete	SecretaireScol	%	*F2B8A4D70CA562E39C10CBBB7CB98E2367980D5F

⇝ utilisateur ajouté, stockage d'une version hachée du mot de passe

Syntaxe de renommage

```
RENAME USER 'AncienNom' TO 'NouveauNom'
```

Syntaxe de changement de mot de passe

```
SET PASSWORD FOR 'Nom' = PASSWORD('NouveauMotDePasse')
```

Plan

Introduction

Data Query Language

Data Manipulation Language

Data Definition Language

Data Control Language

- Utilisateurs
- Attribuer et retirer des droits

Attribuer des droits : GRANT...ON...TO...

Syntaxe générale

```
GRANT NaturePrivilege1,NaturePrivilege2,... ON CibleDuPrivilege TO 'Nom'
```

- CibleDuPrivilege :
  - * : application des droits _____.
  - *.* : application des droits _____
  - NomBase.* : application des droits _____ de NomBase __ _____
  - NomBase.NomTable : application des droits _____ NomBase.NomTable
- NaturePrivilege : ensemble d'autorisations parmi
  - CREATE, ALTER, DROP : _____ _____
  - SELECT, INSERT, UPDATE, DELETE : _____

Exemple : ajout de droits de manipulation de données pour 'SecretaireScol' sur toutes les tables de la base 'APOGEE'

```
GRANT SELECT, INSERT, UPDATE, DELETE ON APOGEE.* TO 'SecretaireScol'
```

Retirer des droits

Syntaxe générale

```
REVOKE NaturePrivilege1,... ON NiveauPrivilege FROM 'Nom'
```

`CibleDuPrivilege` et `NaturePrivilege` : comme dans l'attribution de droits

Exemple : modification structure de la base de données `PROJET` plus autorisée pour l'étudiante de L2 `'Ibrahim'`

```
REVOKE CREATE, ALTER, DROP ON PROJET.* FROM 'Ibrahim'
```

Exemple : interdire l'écriture dans la table `PROJET.LOG` à l'étudiante `'Ibrahim'`

```
REVOKE INSERT, UPDATE, DELETE ON PROJET.LOG FROM 'Ibrahim'
```