Plan

Plan

Ajout de données avec VALUES

Syntaxe

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

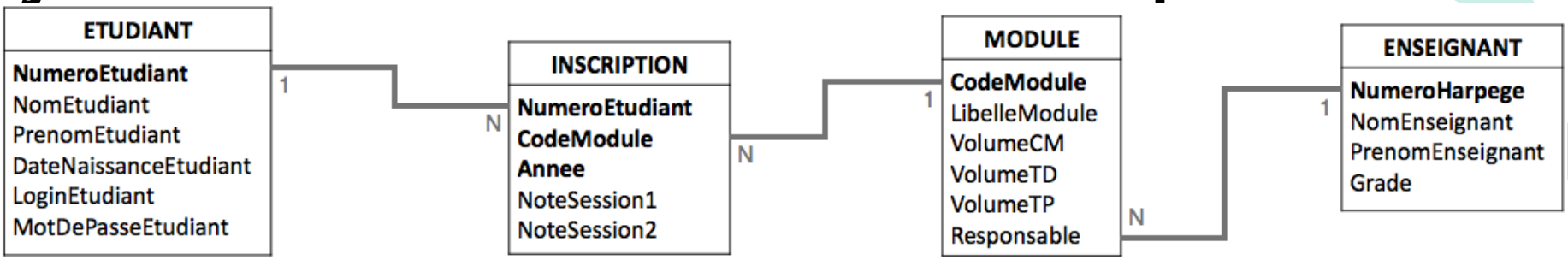
Liste facultative d'attributs `(Att1, Att2, ...)`

- 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

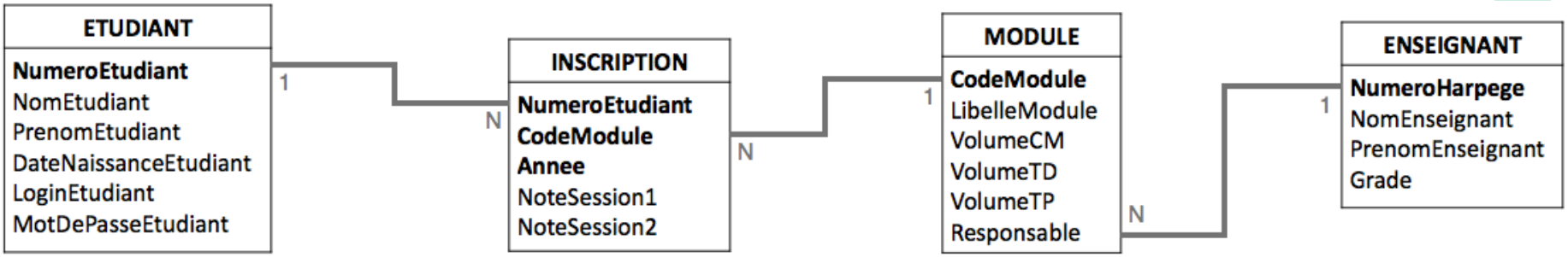
Syntaxe

```
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

Modification de données avec UPDATE : définition

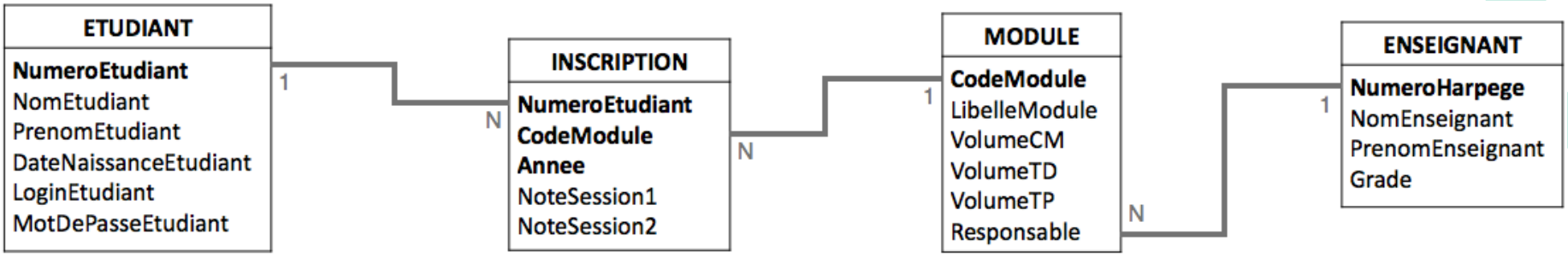
Syntaxe

```
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

Suppression avec DELETE : définition

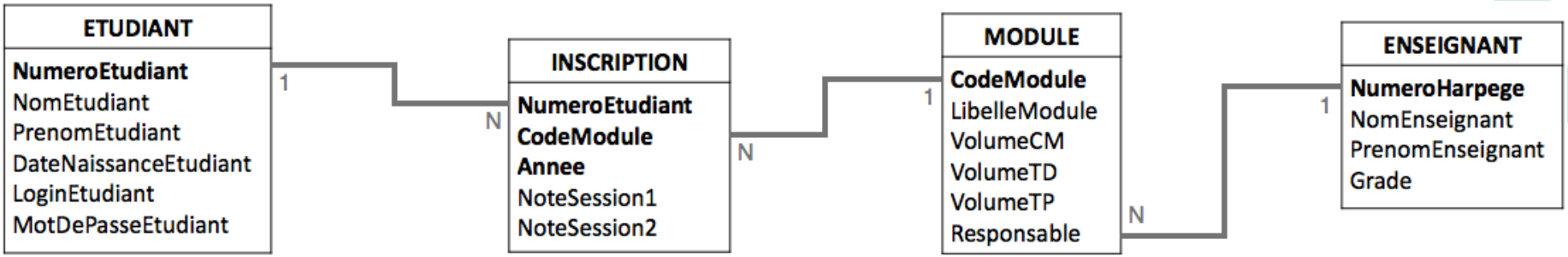
Syntaxe

```
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

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

Plan

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 créat° 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

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

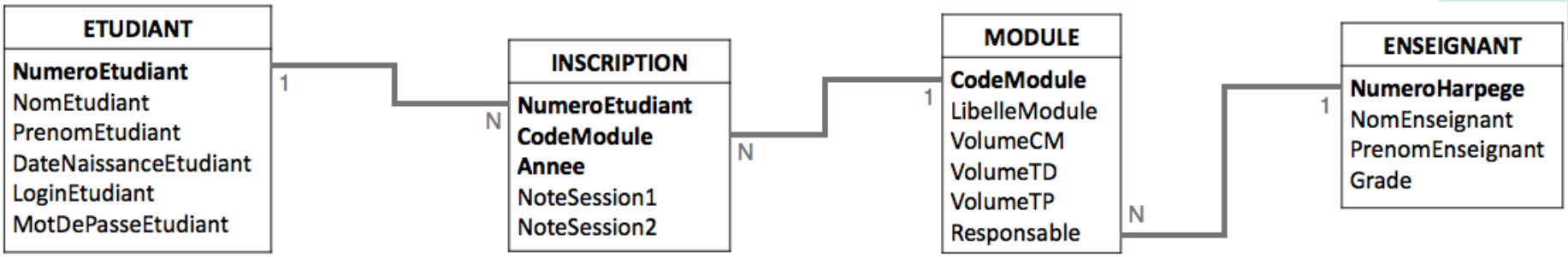
```
CREATE TABLE NOMTABLE
(Att1 Type1 [CA11 ... ],...,
[CONSTRAINT nom_contrainte_table_1] CONTRAINTE_TABLE_1,...,
INDEX [UNIQUE] nom_index (Attj,Attk,...),...)
```

- `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))
```

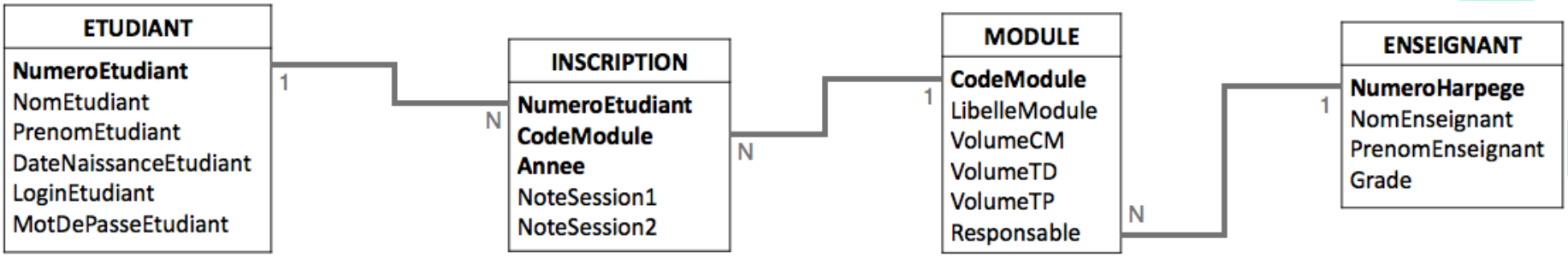
Création : organisation



Préliminaires

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

Création : ENSEIGNANT et MODULE



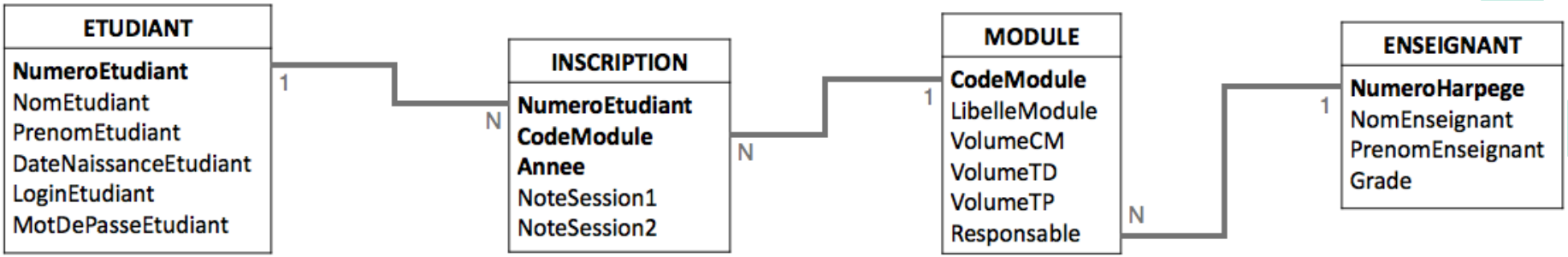
ENSEIGNANT : attributs, clef, index

```
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

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

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

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'
```