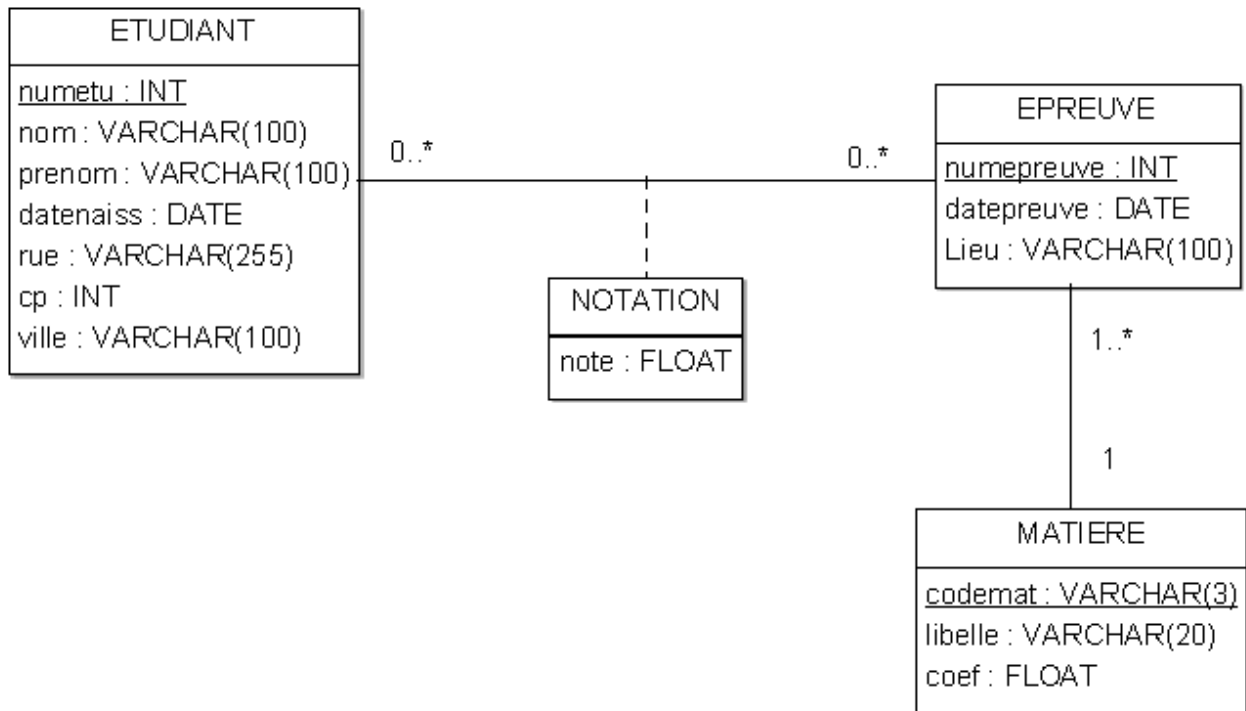


## Base de données

Soit la base de données « notation des étudiants » dont le **modèle conceptuel** est donné ci-dessous sous la forme d'un diagramme de classes UML.



La traduction du modèle conceptuel UML en **modèle logique** relationnel est donnée ci-dessous. Les attributs soulignés sont des clés primaires, ceux suivis d'un # des clés étrangères.

etudiant (numetu, nom, prenom, datenaiss, rue, cp, ville)  
 matiere (codemat, libelle, coef)  
 epreuve (numepreuve, datepreuve, lieu, codemat#)  
 notation (numetu#, numepreuve#, note)

## Création de la base de données

1. Télécharger sur le serveur phpetu, à la racine de votre compte, le fichier MYSQL.<chaîne aléatoire>.txt qui contient les informations de connexion au serveur MariaDB (adresse du serveur, login, mot de passe, nom de la base de données).
2. Se connecter à phpMyAdmin (interface web écrite en PHP qui permet de gérer des bases de données MariaDB) à l'URL <https://phpetu.univ-lyon2.fr/phpmyadmin/> à l'aide de votre login et de votre mot de passe.
3. Dans la partie gauche de l'écran de phpMyAdmin, sélectionner votre base de données.
4. Télécharger le fichier <https://eric.univ-lyon2.fr/jdarmont/docs/td5.sql>, puis utiliser l'onglet « Importer » pour insérer les tables et les données qu'il contient dans votre base.
4. Dans la partie gauche de l'écran de phpMyAdmin, vous devez maintenant pouvoir consulter la structure (**modèle physique**) et le contenu des tables que vous venez d'importer.

## Connexion à la base de données

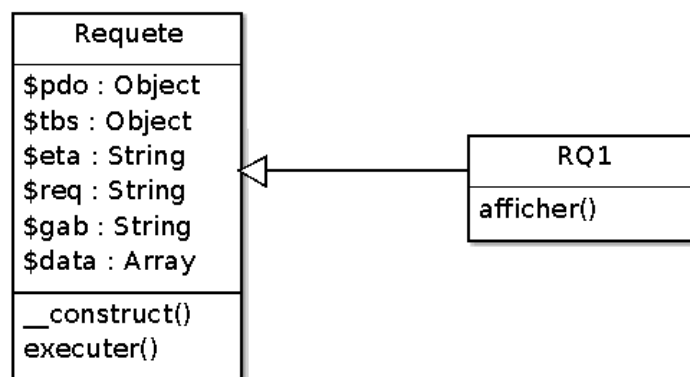
- 1 Créer un fichier PHP nommé connect.inc.php contenant uniquement l'affectation des valeurs suivantes aux variables listées ci-dessous. Ne pas oublier les balises PHP.

```
$host = "adresseServeur";           // ex. mysql02.univ-lyon2.fr
$login = "login";                   // ex. php_jdarmont
$password = "motDePasse";           // cf. fichier MYSQL.<chaîne aléatoire>.txt
$dbname = $login;                   // Normalement, la BD a le même nom que votre login.
```

- 2 Créer un gabarit nommé td5vue.tpl.html sur la base du squelette HTML5 déjà employé aux TD précédents<sup>1</sup>. Dans un paragraphe HTML, insérer le champ [onshow.etatConnexion].
- 3 Créer un fichier nommé td5controleur.php. Y placer les balises PHP, puis inclure les fichiers connect.inc.php et tbs\_class.php. Créer un nouvel objet TinyButStrong, ainsi qu'un nouvel objet PDO (*PHP Data Object*) permettant de vous connecter à MariaDB (T92) en utilisant les variables initialisées dans connect.inc.php. Tester. Si rien ne s'affiche, tout va bien ! (Sinon, il y aurait un message d'erreur.)
- 4 Ajouter au fichier td5controleur.php des instructions de gestion d'erreur (try/catch – T92, T98) incluant les objets TinyButStrong et PDO. En cas de connexion réussie, affecter (toujours dans la partie « try ») à la variable \$etatConnexion (cf. gabarit td5vue.tpl.html) un message du type « Connexion réussie ». En cas d'erreur (partie « catch »), affecter à la variable \$etatConnexion l'erreur standard renvoyée par la méthode getMessage().
- 5 Après la séquence try/catch, afficher le gabarit td5vue.tpl.html. Tester en introduisant des erreurs dans le nom du serveur, de la base de données et dans le couple (login, mot de passe) dans le fichier connect.inc.php, puis rétablissez la connexion correctement.
- 6 Juste après la création de l'objet PDO (connexion à la base), ajouter à ce dernier l'attribut ATTR\_ERRMODE (T98) permettant de traiter les erreurs de requête SQL comme des exceptions. Cela pourrait être utile pour la suite.

## Interrogation de la base de données

Résultat attendu : <https://eric.univ-lyon2.fr/jdarmont/docs/web/td5controleur.php>.



- 1 Créer un nouveau fichier nommé td5modele.class.php. Y créer une classe Requete contenant les attributs **protégés** :
  - \$pdo (un identifiant PDO),
  - \$tbs (un objet TinyButStrong),
  - \$req (une chaîne de caractères contenant une requête SQL),
  - \$gab (une chaîne de caractères contenant un nom de gabarit).Créer un constructeur permettant d'initialiser ces attributs via des paramètres. NB : Les attributs sont tous protégés car la classe Requete ne sera jamais instanciée (c'est une classe abstraite). Seules ses sous-classes vont avoir des objets. Le rôle de la classe Requete est de rassembler des méthodes communes à toutes les sous-classes.

<sup>1</sup> <https://eric.univ-lyon2.fr/jdarmont/docs/web/squelette.html>

- 2 Inclure le fichier `td5modele.class.php` dans le fichier `td5controleur.php`, juste après `tbs_class.php`. Tester. Si le message « Connexion réussie » ne s'affiche pas, c'est qu'il y a une erreur de syntaxe dans `td5modele.class.php`.
- 3 Dans le fichier `td5modele.class.php`, ajouter à la classe `Requete` un attribut protégé `$data` et une méthode publique `executer()` qui, à l'aide de la connexion `$this->pdo`, prépare la requête `$this->req` (T93), l'exécute et stocke toutes les lignes retournées dans `$this->data` (utiliser la méthode `fetchAll` sans le paramètre `PDO::FETCH_ASSOC` – T94).
- 4 Dans le gabarit `td5vue.tpl.html`, définir après le paragraphe d'état de connexion un tableau HTML dont la légende (caption) est « Liste des matières » et dont les données (blocs `<td> </td>`) sont les champs `[codemat.val;block=tr]`, `[libelle.val;block=tr]` et `[coef.val;block=tr]` (T55, T103).
- 5 Dans le fichier `td5modele.class.php`, ajouter une nouvelle classe `RQ1` qui hérite de `Requete`. Cette classe ne comporte qu'une méthode publique nommée `afficher()`, qui prépare les données et affiche le gabarit `td5vue.tpl.html` modifié (T105).
- 6 Dans le script `td5controleur.php`, commenter l'affichage du gabarit `td5vue.tpl.html`, désormais inutile.
- 7 Dans le script `td5controleur.php`, créer un objet `$qmat` de classe `RQ1` avec comme paramètres l'identifiant de connexion PDO, l'objet `TinyButSTrong`, l'état de connexion, la requête SQL qui permet d'obtenir la liste de toutes les matières avec tous leurs attributs descriptifs, et enfin le nom du gabarit (`td5vue.tpl.html`).
- 8 Appeler la méthode `executer()`, puis la méthode `afficher()` pour `$qmat`. Tester. Noter que, dans la classe `RQ1`, la variable `$etatConnexion` n'est pas initialisée ni transmise de l'extérieur... et cela fonctionne tout de même. La raison est que `TinyButSTrong` ne prend en compte que des variables globales soit, dans notre cas, la variable `$etatConnexion` du script `td5controleur.php`. De manière un peu surprenante, les tableaux de valeurs ne fonctionnent pas de la même manière. À vous de réfléchir pourquoi !
- 9 Introduire une erreur dans la requête et vérifier que le `catch()` la détecte bien. Si ce n'est pas le cas, déplacer la création de `$qmat` et l'appel à `executer()` dans le bloc « try » ou revoir la question 7.
- 10 Vérifier la validité du code HTML généré dans votre page `td5controleur.php`.

### **Pour quelques requêtes de plus...**

On veut maintenant afficher la liste des noms et prénoms des étudiant·es, ainsi que leurs notes dans chaque matière (en indiquant le libellé de matière).

- 1 Sur le modèle du gabarit `td5vue.tpl.html`, créer un gabarit `td52vue.tpl.html` permettant d'afficher les champs `nom`, `prenom`, `note` et `libelle` (cf. schéma de la base de données).
- 2 Dans le fichier `td5modele.class.php`, créer une classe `RQ2` sur le modèle de la classe `RQ1` pour préparer les données à l'affichage du gabarit `td52vue.tpl.html`, puis effectuer cet affichage.
- 3 Dans le fichier `td5controleur.php`, commenter les trois lignes de code relatives à la requête `$qmat` puis, sur le modèle de `$qmat`, créer une requête `$qnot` de classe `RQ2`, l'exécuter et l'afficher.
- 4 N'hésitez pas à poursuivre avec d'autres requêtes !

## Correction

```
<?php // connect.inc.php
    $host = "mysql02.univ-lyon2.fr";
    $login = "php_jdarmont";
    $password = "XXX";
    $dbname = $login;
?>
```

```
<!DOCTYPE html> <!-- td5vue.tpl.html -->
<html lang="fr">
    <head>
        <title>Interrogation de base de données</title>
        <meta charset="utf-8" />
    </head>
    <body>
        <p>[onshow.etatConnexion]</p>
        <table>
            <caption>Liste des matières</caption>
            <tr>
                <td>[codemat.val;block=tr]</td>
                <td>[libelle.val;block=tr]</td>
                <td>[coef.val;block=tr]</td>
            </tr>
        </table>
    </body>
</html>
```

```
<?php // td5modele.class.php
```

```
class Requete {
    protected $pdo; // Identifiant de connexion
    protected $tbs; // Moteur de template
    protected $req; // Requête SQL
    protected $gab; // Nom de gabarit
    protected $data; // Résultat de requête

    function __construct($param_pdo, $param_tbs, $param_req, $param_gab) {
        $this->pdo = $param_pdo;
        $this->tbs = $param_tbs;
        $this->req = $param_req;
        $this->gab = $param_gab;
    }

    public function executer() {
        $res = $this->pdo->prepare($this->req);
        $res->execute();
        $this->data = $res->fetchAll();
    }
}
```

```

class RQ1 extends Requete {
    public function afficher() {
        // Préparation des données
        $i = 0;
        $listeCode = array();
        $listeLibe = array();
        $listeCoef = array();
        foreach($this->data as $ligne) {
            $listeCode[$i++] = $ligne["codemat"];
            $listeLibe[$i++] = $ligne["libelle"];
            $listeCoef[$i++] = $ligne["coef"];
        }
        // Affichage du gabarit
        $this->tbs->LoadTemplate($this->gab);
        $this->tbs->MergeBlock("codemat", $listeCode);
        $this->tbs->MergeBlock("libelle", $listeLibe);
        $this->tbs->MergeBlock("coef", $listeCoef);
        $this->tbs->Show();
    }
}

```

```

class RQ2 extends Requete {
    public function afficher() {
        // Préparation des données
        $i = 0;
        $listeNom = array();
        $listePre = array();
        $listeNot = array();
        $listeLib = array();
        foreach($this->data as $ligne) {
            $listeNom[$i++] = $ligne["nom"];
            $listePre[$i++] = $ligne["prenom"];
            $listeNot[$i++] = $ligne["note"];
            $listeLib[$i++] = $ligne["libelle"];
        }
        // Affichage du gabarit
        $this->tbs->LoadTemplate($this->gab);
        $this->tbs->MergeBlock("nom", $listeNom);
        $this->tbs->MergeBlock("prenom", $listePre);
        $this->tbs->MergeBlock("note", $listeNot);
        $this->tbs->MergeBlock("libelle", $listeLib);
        $this->tbs->Show();
    }
}

```

?> }

```

<!DOCTYPE html> <!-- td5vue.tpl.html -->
<html lang="fr">
  <head>
    <title>Interrogation de base de données</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <p>[onshow.etatConnexion]</p>
    <table>
      <caption>Liste des notes</caption>
      <tr>
        <td>[nom.val;block=tr]</td>
        <td>[prenom.val;block=tr]</td>
        <td>[note.val;block=tr]</td>
        <td>[libelle.val;block=tr]</td>
      </tr>
    </table>
  </body>
</html>

<?php // td5controleur.php
require("connect.inc.php");
require("tbs_class.php");
require("td5modele.class.php");

// Moteur de template
$tbs = new clsTinyButStrong;

try {
  // Connexion
  $c = new PDO("mysql:host=$host;dbname=$dbname", $login, $password);
  $etatConnexion = "Connexion OK";
  $c->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  // Liste des matières
  $qmat = new RQ1($c, $tbs, "SELECT * FROM matiere", "td5vue.tpl.html");
  $qmat->executer();
  $qmat->afficher();

  // Liste des notes (exercice complémentaire)
  /* $qnot = new RQ2($c, $tbs, SELECT nom, prenom, note, libelle
  FROM etudiant e, notation n, epreuve p,
  matiere m WHERE e.numetu = n.numetu AND
  n.numepreuve = p.numepreuve AND
  p.codemat = m.codemat", "td5vue.tpl.html");

  $qnot->executer();
  $qnot->afficher(); */

} catch(PDOException $erreur) {
  $etatConnexion = "Erreur : " . $erreur->getMessage();
}

// Affichage état connexion
/* $tbs->LoadTemplate("td5vue.tpl.html");
$tbs->Show(); */

?>

```