I.U.T. LUMIÈRE

UNIVERSITÉ LUMIÈRE LYON 2

Bases de Données Avancées – Module B

IUT Lumière, License CE-STAT
2006-2007
Pierre Parrend

Conception de Systèmes de Bases de Données Avec UML
TD 1 - Correction

Etapes de conception d'un système de base de données:

- Analyse des scénarios
- Réalisation du diagramme de Cas d'Utilisation
  - identification des acteurs, systèmes, et cas d'utilisation
  - réalisation du diagramme de cas d'utilisation globale
  - raffinement des cas d'utilisation: vue systématique des interactions utilisateurs/système
- Réalisation du diagramme de Classes
  - réalisation du diagramme de classe globale
  - identification des classes d'exécution / classes de données
  - raffinement des types de données (pour les classes de données), identification des clés primaires, secondaires
- Réalisation du schéma relationnel
  - rédaction du schéma relationnel représentant les classes
  - normalisation
  - ré-écriture du diagramme de classes si besoin
- Rédation des requêtes SQL
  - création
  - requêtes paramétrées

I. Diagramme de Cas d'Utilisation

Le diagramme de Cas d'Utilisation a pour objectif d'identifier les interactions entre les utilisateurs et le système.

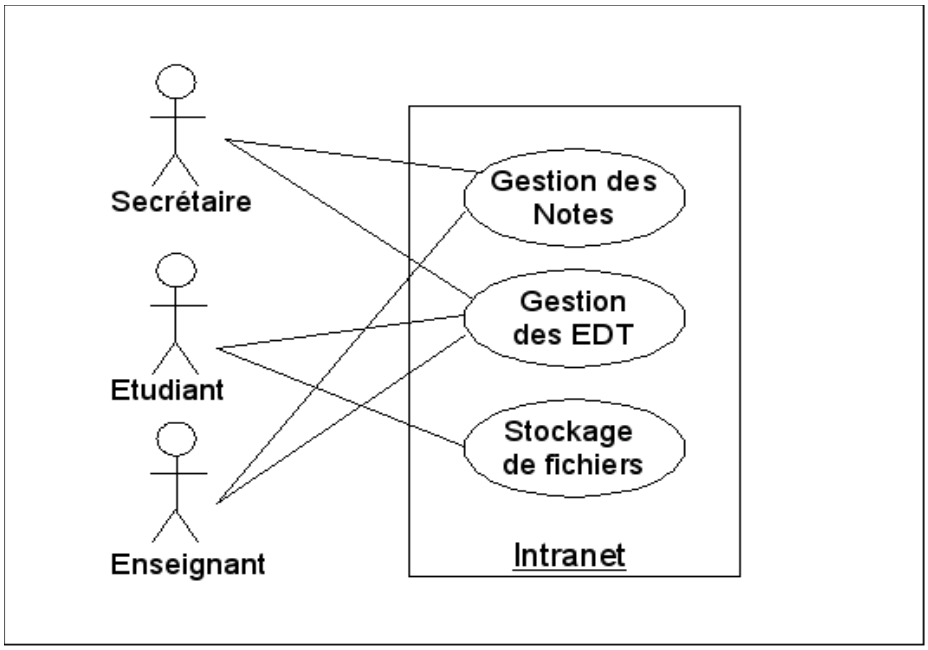
A. Analyse

Acteurs:
- secrétaires
- étudiants
- enseignants

Système(s):
- Serveur Intranet de l'IUT

Cas d'utilisation:
- gestion de l'emploi du temps
- gestion des notes
- stockage de fichier

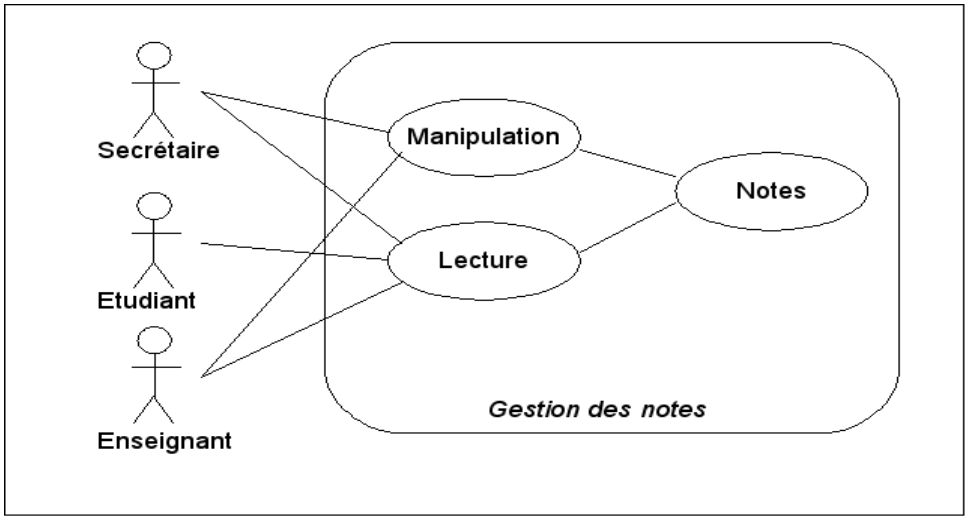
B. Schéma général



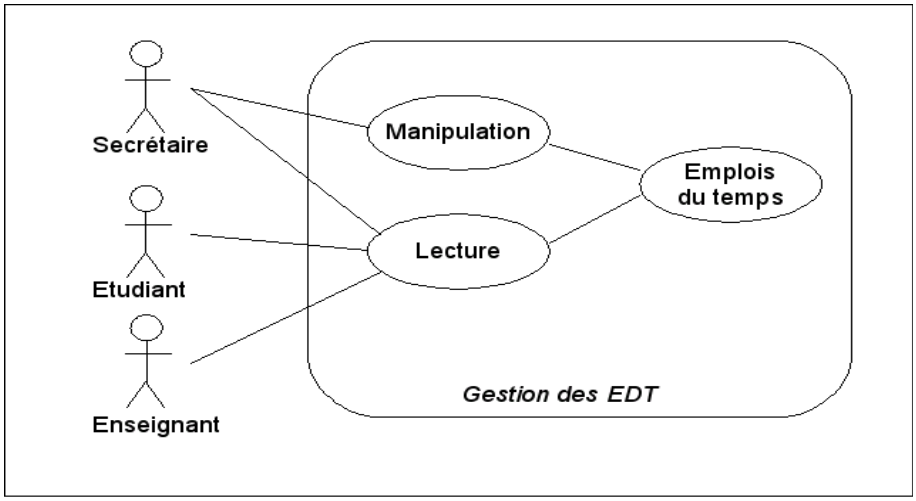
C. Raffinement

The schéma précédent n'indique pas le type d'interaction entre les acteurs et les cas d'utilisations du système. Dans deux cas, les actions possibles dépendent du rôle des acteurs:
- pour la gestion des notes, les étudiants ont un droit de lecture mais pas d'écriture; les secrétaires ont le droit d'écriture sur l'ensemble des données; les enseignants ont le droit d'écriture sur leur matière, et le droit de lecture sur l'ensemble des données
- pour la gestion des emplois du temps, les secrétaires ont le droit de lecture et d'écriture; les étudiants et les enseignants ont le droit de lecture seulement.

Gestion des Notes:



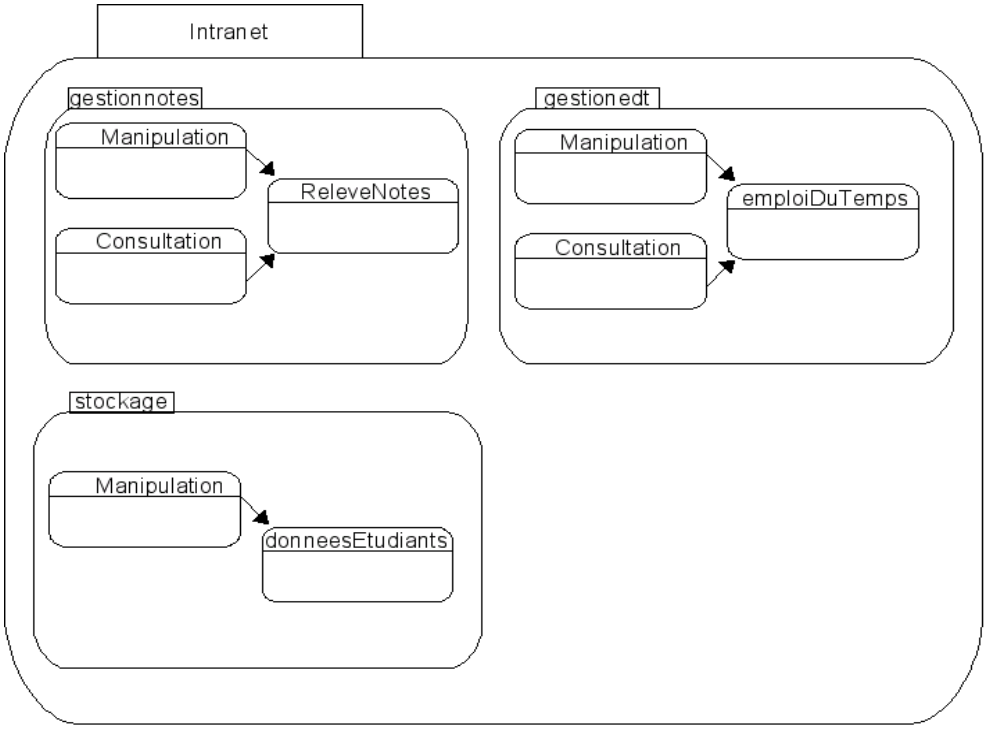
Gestion des Emplois du temps:



II. Diagramme de Classes

Le diagramme de Classes a pour objectif de représenter finement l'architecture interne du logiciel, ainsi que de distinguer les classes d'exécution des classes de données.

A. Diagramme de classes global



B. Classes d'exécution et classes de données

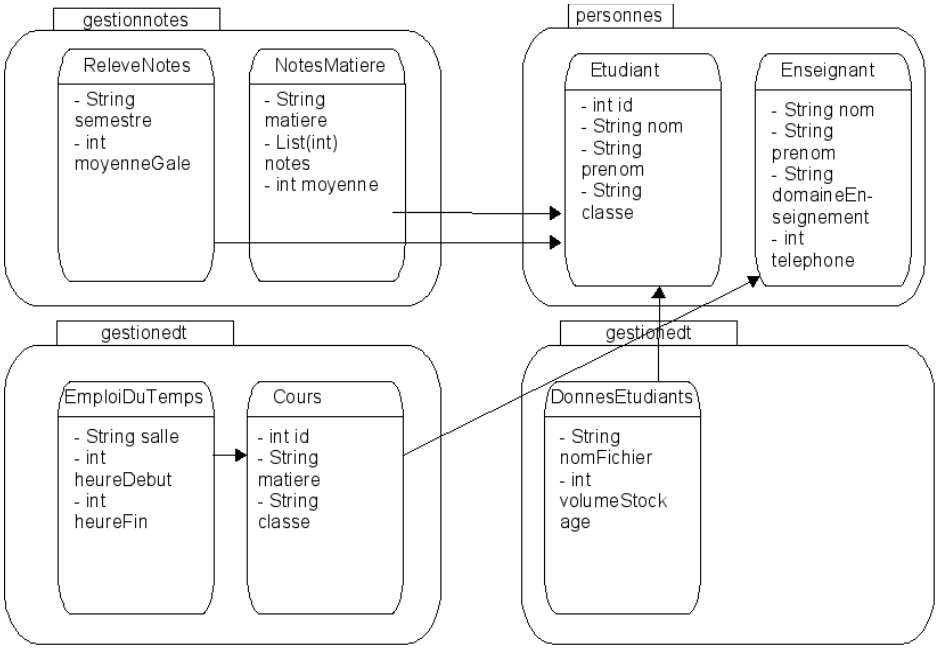
Les classes de données identifiées dans le Diagramme de Classes global sont les suivantes:
- releveNotes
- emploi du temps
- donneesEtudiants

Les classes d'exécutions sont les suivantes:
- gestionnotes.Manipulation
- gestionnotes.Consultation
- gestionedt.Manipulation
- gestionedt.Consultation
- stockage.Manipulation

C. Attributs des classes

Diagramme de classe pour les classes de Données.
Remarques:
- l'identification des attributs des classes permet d'identifier les classes de données complémentaires qui doivent être réalisées,
- Les associations entre classe en UML permettent de ne pas exprimer de manière explicite les liens vers les classes, c'est à dire les clés étrangères. Ces clés devront être inégrées au schéma relationnel.

Le schéma suivant représente les classes de données avec leurs attributs.



D. Clés primaires, clés étrangères

Les clés primaires des futures tables sont les suivantes:

- gestionnotes. releveNotes
  - semestre, Etudiant
- gestionnotes.NotesMatiere
  - matiere, Etudiant
- gestionedt.EmploiDuTemps
  - salle, Cours, heureDebut
- gestionedt.Cours
  - id
- stockage.DonneesEtudiants
  - nomFichier, Etudiant
- personnes.Etudiants
  - id
- personnes.Enseignant
  - nom, prenom

Les clés étrangères sont les suivantes:

- gestionnotes.ReleveNotes
  - Etudiant (clé primaire de la table Etudiant)
- gestionnotes.NotesMatiere
  - Etudiant (clé primaire de la table Etudiant)
- gestionedt.EmploiDuTemps
  - Cours (clé primaire de la table Cours)
- gestionedt.Cours

  - Enseignant (clé primaire de la table Enseignant)
- stockage.DonneesEtudiants
  - Etudiant (clé primaire de la table Etudiant)
- personnes.Etudiants
  - /
- personnes.Enseignant
  - /
- stockage.DonneesEtudiants
  - Etudiant (clé primaire de la table Etudiant)

Les élements suivants ont été identifiés:
- types de données pour l'ensemble des attributs
- clés primaires
- clés secondaires

Les propriétés suivantes sont vérifiées:
- pas de cycles entre les classes
- pas de clés étrangères comportant plus de deux attributs (limite arbitraire de complexité)

Il est donc possible d'établir les schémas relationnels pour les classes de données identifiés

III. Schéma relationnel

A. Schéma relationnel des tables de données identifiées

Cas particulier:
- les listes de données (exemple: les notes) doivent être stockées dans une table spécifique

Remarque:
- les types de données Object (ex.: String, classes) doivent être convertis en types de données relationels (ex.: varchar, clés étrangères)
- les clés primaires et étrangères doivent être indiquées

gestionnotes. releveNotes(varchar semestre, #int idEtudiant, int moyenneGale)

gestionnotes.NotesMatiere(varchar matiere, #int idEtudiant, int moyenne)

gestionnotes.NotesDetail(varchar matiere, #int idEtudiant, int note)

gestionedt.EmploiDuTemps(varchar salle, #int idCours, int heureDebut, int heureFin)

gestionedt.Cours(int id, varchar matiere, varchar classe, #varchar nomEnseignant, #varchar prenomEnseignant)

stockage.DonneesEtudiants(varchar nomFichier, #int idEtudiant, int volumeStockage)

personnes.Etudiants(int id, varchar nom, varchar prenom, varchar classe)

personnes.Enseignant(varchar nom, varchar prenom, varchar domaineEnseignement, int telephone)

B. Normalisation

Les problématiques de normalisation seront étudiées ultérieurement dans le cours.

C. Diagramme de classe raffinés

Une premier diagramme de classe raffiné a été réalisé lors de l'étude des attributs des classes.
L'étude des clés a montré que les critères d'absence de cycle et de faible complexité des clés étrangères sont respectés.
L'étude de normalisation n'a pas été encore abordée dans le cadre du cours.

Par conséquent, le diagramme de classe raffiné est identique au diagramme de classes de données comprenant les attributs

IV. Requêtes SQL

A. Création des tables

Les requêtes SQL de création des tables sont les suivantes:

```
CREATE TABLE ReleveNotes(
        semetre varchar,
        int idEtudiant FOREIGN KEY REFERENCES Etudiants(id),
        int moyenneGale,
        CONSTRAINT Cle_Primaire PRIMARY KEY (semetre, idEtudiant)
);

CREATE TABLE NotesMatiere(
        matiere varchar,
        idEtudiant FOREIGN KEY REFERENCES Etudiants(id),
        moyenne int,
        CONSTRAINT Cle_Primaire PRIMARY KEY (matiere, idEtudiant)
);

CREATE TABLE NotesDetail(
        matiere varchar,
        idEtudiant FOREIGN KEY REFERENCES Etudiants(id),
        note int,
        CONSTRAINT Cle_Primaire PRIMARY KEY (matiere, idEtudiant, note)
);

CREATE TABLE EmploiDuTemps(
        salle varchar,
        idCours int FOREIGN KEY REFERENCES Cours(id),
        heureDebut int,
        heureFin int,
        CONSTRAINT Cle_Primaire PRIMARY KEY (salle, idCours, heureDebut)
);

CREATE TABLE EmploiDuTemps(
        id int PRIMARY KEY,
        matiere varchar,
        classe varchar,
        nomEnseignant varchar,
```

```
        prenomEnseignant #varchar,
CONSTRAINT Cle_Secondaire FOREIGN KEY (nomEnseignant, prenomEnseignant) REFERENCES
Enseignant (nom, prenom)
);

CREATE TABLE DonneesEtudiants(
        nomFichier varchar,
        idEtudiant int FOREIGN KEY REFERENCES Etudiants(id),
        volumeStockage int,
CONSTRAINT Cle_Primaire PRIMARY_KEY (nomFichier, idEtudiant)
);

personnes.Etudiants(int id, varchar nom, varchar prenom, varchar classe)
CREATE TABLE Etudiants(
        id int PRIMARY KEY,
        nom varchar,
        prenom varchar,
        classe varchar
);

personnes.Enseignant(varchar nom, varchar prenom, varchar domaineEnseignement, int telephone)
CREATE TABLE Enseignant(
        nom varchar,
        prenom varchar,
        domaineEnseignement varchar,
        telephone int,
CONSTRAINT Cle_Primaire PRIMARY_KEY (nom, prenom)
);
```

B. Requêtes paramétrées

Les interactions entre les utilisateurs et le systèmes, telles qu'elles ont été identifiées dans le diagramme de Cas d'utilisation, sont les suivantes:

gestionnotes. releveNotes(varchar semestre, #int idEtudiant, int moyenneGale)
gestionnotes.NotesMatiere(varchar matiere, #int idEtudiant, int moyenne)
gestionnotes.NotesDetail(varchar matiere, #int idEtudiant, int note)
gestionedt.EmploiDuTemps(varchar salle, #int idCours, int heureDebut, int heureFin)
gestionedt.Cours(int id, varchar matiere, varchar classe, #varchar nomEnseignant, #varchar prenomEnseignant)
stockage.DonneesEtudiants(varchar nomFichier, #int idEtudiant, int volumeStockage)
personnes.Etudiants(int id, varchar nom, varchar prenom, varchar classe)
personnes.Enseignant(varchar nom, varchar prenom, varchar domaineEnseignement, int telephone)

– manipulation de l'emploi du temps par les secrétaires

```
INSERT INTO EmploiDuTemps
        (salle, id, heureDebut, heureFin)
```

– consultation de l'emploi du temps par les enseignants.

Hypothèse: un enseignant recherche les informations qui correspondent aux cours qu'il donne. Le nom de l'enseignant concerné est passé en paramètre, il est noté $ens

Par exemple, m. Martin consulte son emploi du temps

```
Select *
        from EmploiDuTemps INNER JOIN Cours
        USING EmploiDuTemps.idCours = Cours.id
        where Cours.nomEnseignant = $ens
```

– consultation de l'emploi du temps par les étudiants

Hypothèse: les étudiants de la classe $classe (passé en paramètre) consultent leur emploi du temps

```
Select *
        from EmploiDuTemps INNER JOIN Cours
        USING EmploiDuTemps.idCours = Cours.id
        where Cours.classe = '$classe'
```

– manipulation des notes par les secrétaires

Hypothèse: les secrétaires peuvent visualiser les notes, les moyennes, la moyenne générale. Elles doivent pouvoir calculer la moyenne générale d'un étudiant, repéré par son identifiant $id.

```
Select * from NoteMatiere
        where idEtudiant = $id
Select * from ReleveNote
        where idEtudiant = $id
INSERT INTO releveNote (semestre, $id, Select AVG (moyenne) from NoteMatiere where idEtudiant = $id)
```

– manipulation des notes par les enseignants

Hypothèse: les enseignants doivent pouvoir ajouter une note $note, calculer la moyenne de leur matière, et consulter les résultats globaux des étudiants

```
Select * from ReleveNote
        where idEtudiant = $id
INSERT INTO NoteMatiere (semestre, $id, Select AVG (moyenne) from NoteDetail where idEtudiant = $id)
INSERT INTO NoteDetail (semestre, $id, $note)
```

– manipulation des notes par les étudiants

Hypothèse: les étudiants doivent pouvoir consulter leurs résultats

```
Select * from NoteDetail
        where idEtudiant = $id
Select * from NoteMatiere
        where idEtudiant = $id
Select * from ReleveNote
        where idEtudiant = $id
```