Projet
Algo-Prog 5 – J2EE

- Le projet sera rendu sous forme d'archive compressée (jar), au plus tard lors des dernières séances de TD AP5 (la date sera précisée ultérieurement)

A Objectifs

- Il s 'agit de concevoir une application Web permettant la gestion d'emploi du temps, dans un contexte organisationnel comparable à celui du DUT informatique de Caen.

B Contraintes

- L'application sera développée en Java (j2ee),
- Elle respectera au mieux le design pattern MVC2, en séparant modèle (classes Métier), vues (interfaces web de saisie et d'affichage) et contrôleur (vérifiant l'intégrité des données).
- Elle utilisera la base de données Mysql fournie en annexe.
- L'utilisation de librairies ou de FrameWorks supplémentaires, vivement conseillée, est cependant laissée à la libre appréciation de l'étudiant.
- L'utilisation de scripts côté client (javascript et ajax) pourra compléter les validations côté serveur.
- L'application sera constituée de quatre modules indépendant (cf plus loin)

C Règles de gestion :

1 Les cours :

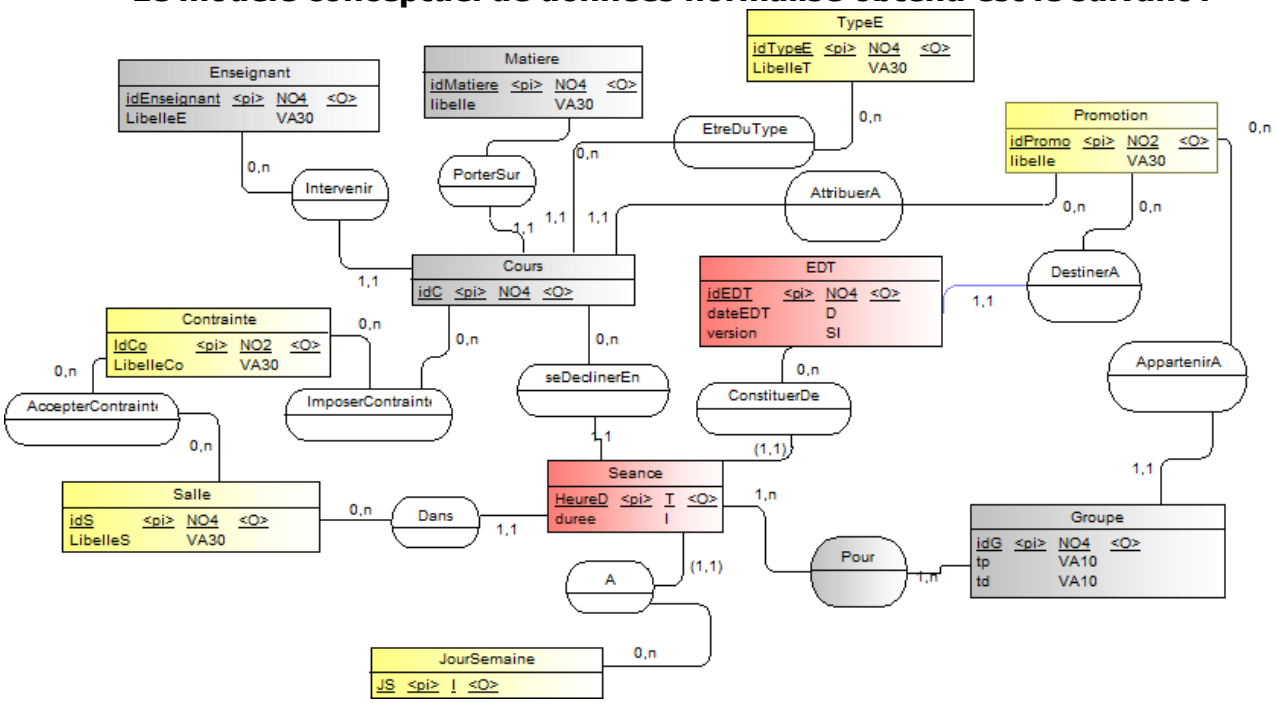
- un **cours** correspond à l'intervention d'un **enseignant** pour une **promotion** dans une **matière**, suivant des modalités d'intervention nommées **TypeE** (CM, TD ou TP par exemple).
- **Exemple de cours :**
- Enseignant : O. Charles – Matière : Qualité – TypeE : TD – promo : info2
- Chaque cours peut imposer certaines **contraintes** : un CM nécessite une salle pouvant accueillir 80 étudiants, un TP d'algo-prog nécessite une salle informatique...
- Les salles peuvent quant-à elles accepter certaines contraintes (l'amphi peut accueillir 80 étudiants, et plus, par exemple)

2 Séances de cours et emploi du temps

Chaque emploi du temps (**Edt**) est actif à partir d'une date, dispose d'une validité (à activer) ,d'un numéro de version et est associé à une promotion.
L'emploi du temps est constitué d'un ensemble de **séance**s de cours.
Une séance correspond à la réalisation d'un cours (trio enseignant-matière-typeE précédemment décrit) dans une salle, un certain jour de la semaine, à partir d'une heure de début et pour une certaine durée.

- **Exemple de séance de cours :**
- Séance : Cours (O. Charles – Qualité – TD - info2) – JourSemaine : Lundi – Salle : 2235 – HeureD : 8:00 – Durée : 6 ¼ d'heure
- Chaque séance est réalisée devant un ou plusieurs groupes d'étudiants :

- **Exemple de Réalisation de séance de cours :**
- Réalisation Séance : Cours (O. Charles – Qualité – TD - info2) – JourSemaine : Lundi – Salle : 2235 – HeureD : 8:00 – Durée : 6 ¼ d'heure → groupes de TP 1.1 et 1.2 (2ème année)
- **Le modèle conceptuel de données normalisé obtenu est le suivant :**



D Dénormalisation :

Afin d'optimiser les accès à la base de données, ainsi que le code produit, le MCD est dénormalisé :

Les contraintes :

L'entité contrainte et ses deux relations n-aires sont supprimées, pour introduire dans l'entité Cours et l'entité Salle un champ **contraintes** multivalué.

- **Exemples de contraintes :**

Sur salle :

- Salle 2235 - Contraintes : **nb=20;type=info**

Sur cours :

- Enseignant : O. Charles – Matière : Qualité – TypeE : TD – Contraintes : **type=info**

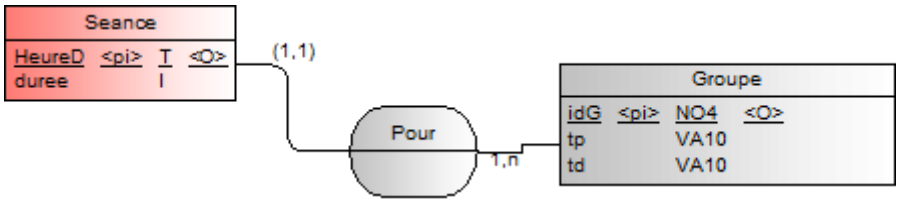
Le champ **contraintes** de chacune des deux tables contiendra des couples clé=valeur séparés par un point-virgule.

Chaque cours devra donc se tenir dans une salle acceptant les contraintes qu'il impose.

Les séances des groupes (association n-aire Pour) :

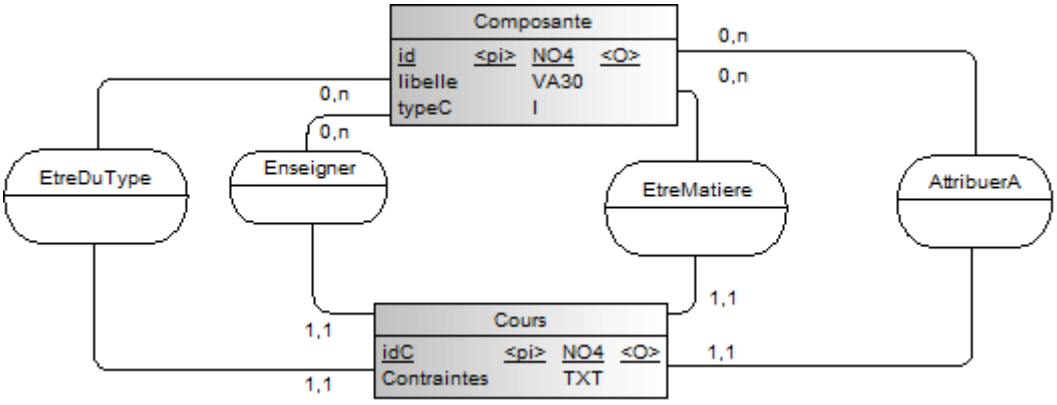
La CIM **« Pour »** est transformée en CIF, avec identification relative du cours par rapport au Groupe.

Pour un même cours de type TD face au Groupe TD 2.2 (2ème groupe de 2ème année), on ajoutera deux occurrences de séance, l'une pour le Groupe de TP 2.2.1, et l'autre pour le groupe de TP 2.2.2

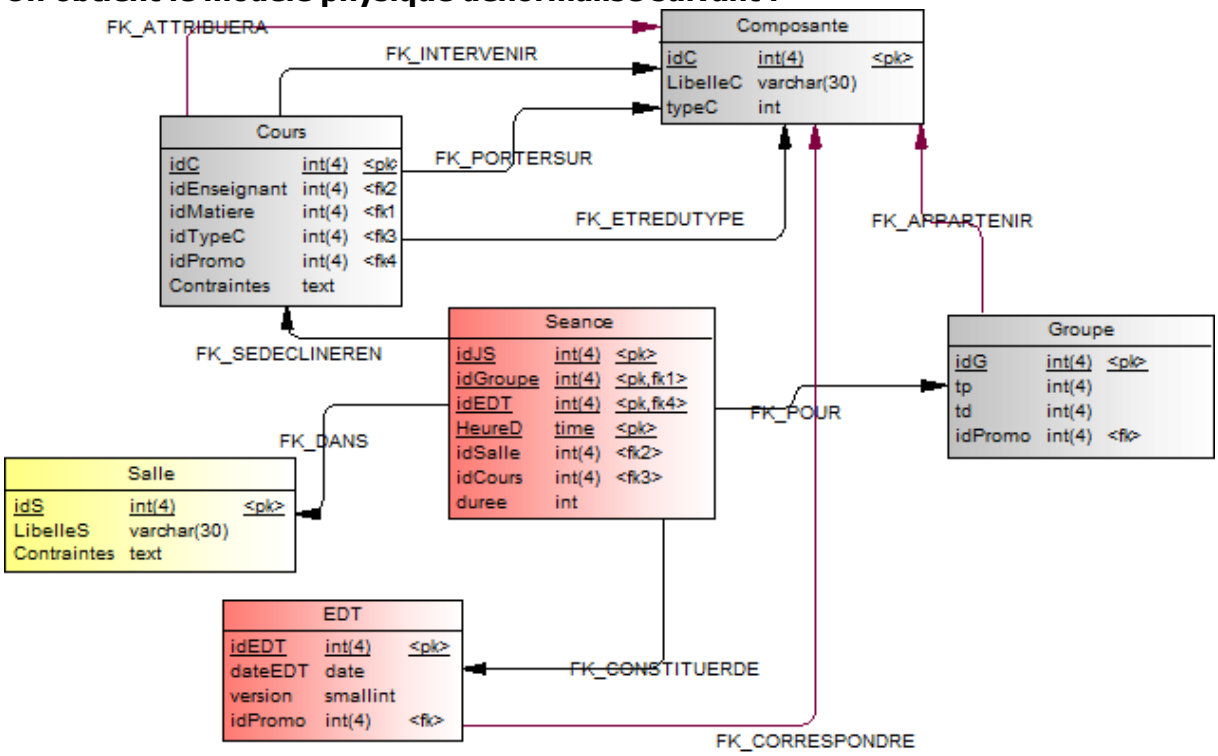


Promotions, Matières, enseignants et types de cours

On peut facilement poursuivre la dénormalisation en factorisant les entités Promotion, Matière, Enseignant et TypeE pour les regrouper dans une Entité « Composante », munie des propriétés (id, libelle et typeC), où la propriété typeC désigne une promotion, un Enseignant, une Matière ou un type de cours.



On obtient le modèle physique dénormalisé suivant :



- **Remarque :**

Il appartient à l'étudiant de choisir le modèle dénormalisé qui lui semble le plus efficace.

E Descriptif de la saisie d'une séance :

La saisie d'une séance dans un emploi du temps devra suivre un scénario comparable à celui-ci :
La création d'une séance consiste à affecter un cours à un groupe d'étudiants, un jour de la semaine, pour une durée, dans une salle.
Le programme ne laissera le choix que des solutions viables, plutôt que d'afficher des messages d'erreur face à un choix impossible.
Cas d'utilisation : Placement/création d'une séance de cours
Acteur principal : Responsable des emplois du temps
Précondition : Il existe un emploi du temps, des cours à placer, des groupes, des jours, des salles créés. La liste des emplois du temps est affichée.
PostCondition : une séance de cours a été créée et placée dans un emploi du temps
Scénario nominal :

1. L'utilisateur sélectionne un emploi du temps
2. Le système affiche les cours à placer ainsi que les groupes d'étudiants correspondant à la promotion de l'emploi du temps
3. L'utilisateur sélectionne le cours à placer
4. Il saisit de la durée de la séance (en quart d'heures)
5. Le système affiche des jours et plages horaires possibles en fonction des disponibilités de l'enseignant (contrainte Enseignant) + contrainte durée
6. L'utilisateur sélectionne un ou plusieurs groupe d'étudiants
7. Le système Actualise les jours et horaires disponibles en fonction des disponibilités des groupes (contrainte groupes)+ contrainte Enseignant + contrainte durée

8. A - L'utilisateur sélectionne un jour	B - L'utilisateur sélectionne une salle
9. A - Le système actualise les horaires disponibles (Contraintes : Durée + Enseignant + groupe + jour) Affichage des salles disponibles	Le système actualise les jours et horaires disponibles (Contraintes : Durée + Enseignant + groupe + Salle)
10. A - L'utilisateur choisit un horaire	B - L'utilisateur choisit un jour
11. A - Le système affiche les salles disponibles en respectant les contraintes du cours + contrainte jour + Durée	B - Le système actualise les horaires disponibles (Contraintes : Durée + Enseignant + groupe + Salle + jour)
12. A - L'utilisateur choisit une salle	B - L'utilisateur choisit un horaire

13. L'utilisateur Valide
14. Le système sauvegarde séance

Scénarii alternatifs :

Des étapes 8 à 12 :

- l'utilisateur peut alterner entre les étapes A et B en fonction des disponibilités retournées par le système
- l'utilisateur peut afficher l'emploi du temps du groupe sélectionné pour déplacer une autre séance.

L'utilisateur peut abandonner à tout moment la saisie

F Modules et fonctionnalités :

Module	Fonctionnalités
DS :Données de structure • Jours de la semaine • Salles • Types d'enseignement (TypeE)	• Ajout/modification/suppression • Ajout/modification/suppression/gestion des contraintes • Ajout/modification/suppression/ [Contraintes?]
DA : Données annuelles/Semestrielles • Enseignants • Promotions • Groupes d'étudiants • Matières • Cours	• Ajout/modification/suppression/ [disponibilités?] • Ajout/modification/suppression • Ajout/modification/suppression • Ajout/modification/suppression • Ajout/modification/suppression/Contraintes/Duplication/Affichage par groupe
EDT : Conception emplois du temps	• Création et placement de séances de cours, pour le jour, l'heure, le groupe et la salle, contrôle des disponibilités des groupes, salles, enseignants, cohérence entre contraintes salles et cours • Assistance au placement d'une séance • Gestion des modifications entre versions (affichage des changements, envoi de mail aux intervenants)
Diff : Diffusion pour affichage	• Affichage (HTML/pdf) par semaine pour un ou plusieurs groupes donnés • Affichage par enseignant • Affichage disponibilité/occupation salle • Envoi par mail

[fonctionnalité optionnelle envisageable]

G Travail à rendre

Le projet est à réaliser en binômes (**obligatoire**), il sera rendu sous la forme d'une archive zippée nommée **Prenom1Nom1-Prenom2Nom2.zip**, où prenom1Nom1 et prenom2Nom2 représentent les prénoms et noms des 2 étudiants participant.

L'archive devra comprendre :

1. une archive Web compressée des fichiers du projet (.war)
2. une documentation technique composée :
   a. de la javadoc complète des classes, objets, variables, méthodes et fichiers créés
   b. d'un plan détaillé du site précisant les interfaces, les fonctionnalités par interface, et la possible navigation entre elles
   c. d'un bref manuel permettant l'installation de l'application
3. une documentation utilisateur au format html ou pdf permettant la prise en main de l'outil pour les non-initiés.
4. Un aperçu de la répartition des tâches entre les membres du binôme, basé sur le découpage en modules et fonctionnalités (Tableau avec couleurs).

H Critères de notation

- **Le code :**

Il devra être maintenable et évolutif, facilement reprenable par une autre équipe de développement.

- Nommage correct (classes, variables, fichiers, membres...)respectant des normes
- respect du principe de délégation
- respect des principes de la POO
- Séparation des couches (MVC2)
- Lisibilité et documentation du code

- **Les interfaces :**

On veillera particulièrement à leur ergonomie :

- à la simplicité de prise en main (sans documentation),
- au respect des normes de présentation des interfaces web,
- à l'accessibilité des fonctionnalités,
- à la normalisation W3C.

- **L'application :**

- Elle devra implémenter de façon complète et accessible les fonctionnalités des 4 modules décrits, en respectant les contraintes données.