MINI PROJET – GENIE LOGICIEL
SYSTEME INTEGRE DE GESTION DE BIBLIOTHEQUE (SIGB)

Université des sciences et de la technologie Houari Boumediene (USTHB)
3ème année Licence académique
Année universitaire 2009/2010

1 Modalité de déroulement du mini-projet

Ce mini-projet est à réaliser par équipe de 2 ou 3.

- Il s'effectuera du 17 décembre 2009 au 04 janvier 2010.
- Le mini-projet donnera lieu à la remise d'un dossier, dans un format papier, le 04 janvier 2010 dans mon casier avant 13h00 (ou pendant la séance du cours).
- Les diagrammes UML pourront être réalisés avec un simple outil de dessin, un outil UML gratuit ou en version d'évaluation. Pour produire les diagrammes UML avec des outils spécialisés, vous pouvez par exemple utiliser :
  - EclipseUML http://www.eclipsedownload.com/download_studio_eclipse_3x.html
  - Objecteering http://www.objecteering.com/downloads_uml_free_edition.php
  - Poseidon (version « community edition ») http://www.gentleware.com/.
  - Umbrella.
  - ArgoUML
  - Etc[1].

2 Contenu du dossier à rendre

Le dossier comprendra l'analyse et la conception d'un Système Intégré De Gestion De Bibliothèque (SIGB).
Il sera composé des éléments suivants :

- Une brève présentation des processus unifiés, notamment 2TUP (Two Tracks Unified Process) (2 pages maximum).
- L'analyse et la conception en UML, avec les diagrammes de cas d'utilisation, un diagramme détaillé de classes (signature typée des attributs et des méthodes).

3 Expression des besoins

3.1 Description générale du domaine

Une bibliothèque comporte des documents de divers types qui peuvent être prêtés ou consultés sur place. Les documents sont classés selon les règles de la communauté des bibliothèques. Des informations sont associées aux documents en fonction de leur nature, par exemple le titre, le(s) auteur(s), la date de parution, etc.

[1] Voir http://uml.developpez.com/outils/

Les abonnés ayant accès à la bibliothèque ne peuvent emprunter qu'un nombre limité de documents et n'ont pas nécessairement tous les mêmes droits.

3.2 Description des besoins

Il s'agit de réaliser un Système Intégré de Gestion de Bibliothèque (SIGB) d'une organisation qui met des documents à la disposition de ses membres.

Le SIGB répertorie l'ensemble des documents constituant le fonds de la bibliothèque. Ce fonds est constitué actuellement de livres et de périodiques. Il est prévu que le fonds contienne dans l'avenir d'autres catégories de documents.

Chaque document a un titre, une année de publication, un éditeur et une unique référence. Un livre a un (des) auteur (s), et un code ISBN (International Standard Book Number). Un périodique a un volume, un numéro ainsi qu'un code ISSN (International Standard Serial
Number).

Les livres peuvent être des romans ou des manuels. Les romans ont éventuellement un prix littéraire (un entier, parmi : GONCOURT, MEDICIS, INTERALLIE, NOBEL, ACADEMIE), tandis que les manuels ont un niveau scolaire (entier).

Les périodiques sont consultables sur place uniquement et la bibliothèque ne dispose que d'un seul exemplaire de chacun des périodiques. Un livre peut être emprunté ou consulté sur place. S'il existe plusieurs exemplaires d'un livre, chaque exemplaire a une cote qui lui est propre.

Un exemplaire d'un livre présent dans la bibliothèque est dit « en rayon ». Quand un utilisateur emprunte un exemplaire d'un livre, l'exemplaire est dit « en prêt ». Un exemplaire qui n'a pas été rendu dans les délais par un utilisateur est dit « en retard ».

Chaque exemplaire d'un livre dispose d'un état. Les valeurs possibles sont au nombre de 5: *neuf*, *très bon état*, *bon état*, *usagé*, *endommagé*. Si, au retour, il s'avère qu'un livre baisse de plus de 3 niveaux, une amende forfaitaire, à préciser en fonction de la valeur du livre est demandée à l'utilisateur. Les livres devenu obsolète (dans l'état endommagé) sont détruits.

Le système enregistre toutes informations relatives aux utilisateurs de la bibliothèque. Il distingue trois catégories de clients : les *utilisateurs occasionnels* qui ont le droit d'emprunter un seul livre à la fois pour une durée de 15 jours, les *abonnés* qui ont le droit d'emprunter en même temps 3 livres au plus pendant un mois, les *abonnés privilégiés* qui ont le droit d'emprunter au maximum 5 livres simultanément au plus pendant un mois.

Tout emprunt est enregistré dans le système. Chaque emprunt a une durée limitée définie par une date de début et une date de fin. Un emprunt peut être prolongé pour une nouvelle période.

Si un utilisateur ne rend pas dans les délais un exemplaire de livre emprunté, un rappel est envoyé par courrier électronique. Un utilisateur ne peut plus emprunter de livres tant qu'il n'a pas rendu les exemplaires conservés au delà des délais de prêt.
Le système est prévu pour qu'au delà de trois relances à une semaine d'intervalle, le directeur de la bibliothèque soit informé afin qu'il puisse prendre les mesures nécessaires pour obtenir la restitution des exemplaires indûment conservés par l'emprunteur.

Les bibliothécaires forment plusieurs catégories en fonction de leur position hiérarchique. Les *stagiaires* ne sont présents que pour une faible durée. Ils viennent aider les *bibliothécaires principaux* pendant les périodes d'affluence (avant et au retour des vacances scolaires) et ne peuvent qu'enregistrer des emprunts ou des retours. Les *bibliothécaires principaux* peuvent en plus prolonger un emprunt et interdire temporairement à un utilisateur d'emprunter des documents.

4 Travail à réaliser

4.1 Analyse

A. IDENTIFIER LES ACTEURS

Les acteurs candidats sont systématiquement :

- les utilisateurs humains directs;
- les autres systèmes connexes qui interagissent aussi directement avec le système étudié.

Distinguer acteur principal et acteur secondaire :

- l'acteur principal est celui pour qui le cas d'utilisation produit un résultat observable; généralement, c'est lui qui déclenche, le cas d'utilisation.
- Les acteurs secondaires sont les autres participants du cas d'utilisation
  - sollicités pour des informations complémentaires
  - consultent ou surveillent le système

B. IDENTIFIER LES CAS D'UTILISATION

L'ensemble des cas d'utilisation doit décrire exhaustivement les exigences fonctionnelles du système. Chaque cas d'utilisation correspond donc à une fonction du système, selon le point de vue d'un de ses acteurs.
Pour chaque acteur, il convient de :

- rechercher les différentes intentions avec lesquelles il utilise le système,
- déterminer dans le cahier des charges les services fonctionnels attendus du système.

Nommez les cas d'utilisation par un verbe à l'infinitif suivi d'un complément, du point de vue de l'acteur (et non pas du point de vue du système).

C. ORGANISER LES CAS D'UTILISATION

1) Ajouter des relations entre cas d'utilisation:

a) Relation «*include*» entre cas d'utilisation:

i) Le cas de base incorpore explicitement un autre cas.

ii) Le cas inclus n'est jamais exécuté seul, mais uniquement en tant que partie
iii) Cette relation est utilisée pour factoriser un comportement commun présent dans plusieurs cas d'utilisation

b) Relation «*extend*» entre cas d'utilisation
i) Le cas de base incorpore un autre cas, à un endroit spécifié.
ii) Le cas de base peut fonctionner tout seul, ou être aussi complété par un autre.
iii) Utilisé pour séparer le comportement optionnel.

c) de généralisation/spécialisation
i) Les cas d'utilisation peuvent être hiérarchisés par généralisation/spécialisation.
ii) Les cas d'utilisation enfant héritent des comportements de leur parent.

2) Regrouper les cas d'utilisation en Package

Réaliser un diagramme de paquetage regroupant les cas d'utilisation. Dans cette première étape, le regroupement se fait principalement par grandes fonctionnalités, et par priorité de développement ou par acteur (si chaque cas d'utilisation ne fait pas intervenir plusieurs acteurs).

- Un package UML est un espace de nommage qui peut contenir:
  - des éléments d'un modèle
  - des diagrammes qui représentent les éléments du modèle
  - d'autres packages
- Dans un package, on trouve des éléments qui:
  - représentent un ensemble fortement cohérent
  - sont généralement de même nature et de même niveau sémantique.

D. DESCRIPTION TEXTUELLE DES CAS D'UTILISATION

On se limitera dans cette étude à la description textuelle des cas d'utilisation suivants :

- Emprunt d'un livre par un abonné;
- Retour d'un livre emprunté par un abonné ;
- Inscrire un nouvel abonné.

La fiche de description textuelle d'un cas d'utilisation n'est pas normalisée par UML.

Décrire les cas d'utilisation susmentionnés sous la forme du tableau ci-dessous :

Nom	Nom du cas d'utilisation
Résumé	Ce que fait le cas d'utilisation en quelques lignes
Auteur	Nom de l'auteur de cette description
Dates de création	Dates de création de la description
Date de mise à jour	Date de la dernière mise à jour
Acteurs	Acteurs impliqués dans le cas d'utilisation
Pré conditions	Condition sur le déclenchement du cas d'utilisation. Contient aussi les contraintes sur l'état du système avant l'exécution
Post conditions	Etat du système après l'exécution.
Description des scénarios	Décrit le scénario nominal, les scénarios (ou enchaînements) alternatifs, les scénarios (ou enchaînements) d'erreur.
Exigences non fonctionnelles (optionnel)	Ajoute, si c'est pertinent, les informations suivantes : fréquence d'utilisation dans le système, volumétrie, disponibilité, fiabilité, intégrité, confidentialité, Performances (Contraintes temporelles), concurrence, etc. Précise également les contraintes d'interface homme-machine comme des règles d'ergonomie, une charte graphique, etc.

Un cas d'utilisation n'est ni une transaction, ni une fonction: ne pas descendre trop bas en terme de granularité (*difficulté de savoir à quel niveau de détail s'arrêter.*)

Un cas d'utilisation ne doit pas se réduire à une seule séquence ou à une simple action.

E. IDENTIFICATION DES CLASSES CANDIDATES DU MODELE STATIQUE D'ANALYSE

Une des étapes dans l'analyse peut aussi être la réalisation d'une liste d'objets candidats. Cette liste peut être établie en soulignant les noms dans la définition du problème.

Réaliser un premier diagramme de classes qui contiendra les abstractions clés retenues et qui mettra en évidence des associations entre les classes. Les associations seront de simples traits banalisés : pas de nom, pas de multiplicité (cardinalité).

4.2 Conception

A. DIAGRAMME DE CLASSE DE CONCEPTION

Réaliser un diagramme de classe sur le Système Intégré de Gestion de Bibliothèque en partant du diagramme réalisé dans l'étape précédente.

Le nouveau diagramme contiendra des informations complémentaires en termes de caractéristiques des associations (nom, cardinalités[2], **classes d'associations**[3], …).

[2] Vous pouvez réaliser un digramme objet pour valider les cardinalités.
[3] Voir cours UML

- Définir les rôles des classes dans les associations.
- Compléter les classes en termes d'attributs et opérations. Si nécessaire, créer des classes pour définir de nouveaux types.
- Définir les niveaux de visibilités (public, private, protected), les types, les paramètres, les valeurs d'initialisation.
- Reprendre les associations existantes et définir si ce sont de simples associations, des agrégations ou des compositions. Faire éventuellement des restrictions sur le sens de navigation.
- Identifier d'éventuelles classes d'associations permettant de caractériser les associations.
- Identifier les liens d'héritage possibles. Cette tâche fait ajouter des classes au diagramme initial qui permette une généralisation ou une spécialisation de classe.
- Vérifier que le diagramme permet de remplir les différents cas d'utilisation et leurs scénarii associés.
  Pour faire cette vérification, il faut prendre un scénario et suivre les associations sur le diagramme de classes en même temps que les messages sont envoyés. Les associations présentes doivent permettre de « porter » la communication entre les objets.

B. DU DIAGRAMME DE CLASSES UML VERS UN SCHEMA RELATIONNEL

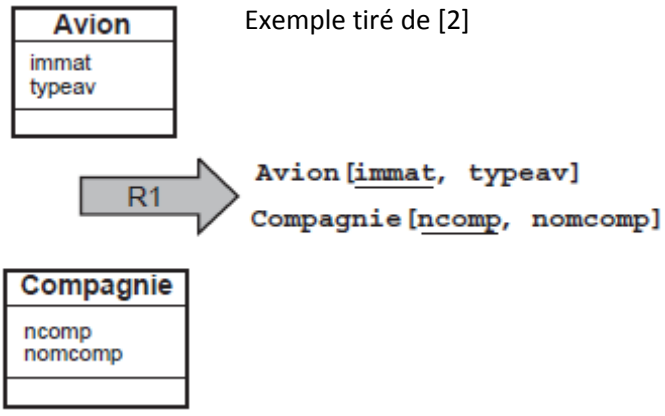
Proposez un schéma relationnel qui correspond au diagramme de classes élaborer lors de l'étape précédente.

Nous donnons ci-après quatre règles (de R1 à R4) pour traduire un schéma conceptuel
UML en un schéma relationnel équivalent [2]. Il existe d'autres solutions de transformation, mais ces règles sont les plus simples et les plus opérationnelles.

Transformation des classes

R1 Chaque classe du diagramme UML devient une relation. Chaque attribut de la classe devient attribut de la relation. Il faut choisir un attribut de la classe pouvant jouer le rôle d'identifiant.
Si aucun attribut ne convient en tant qu'identifiant, il faut en ajouter un de telle sorte que la relation dispose d'une clé primaire (les outils proposent l'ajout de tels attributs).

Exemple tiré de [2]



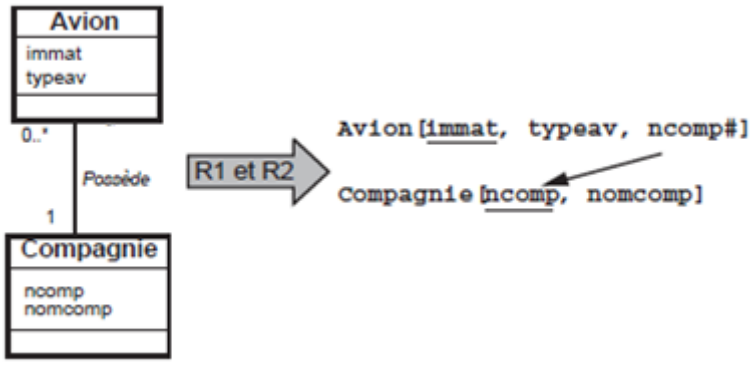
Transformation des associations

Les règles de transformation des associations dépendent des cardinalités/multiplicités maximales des associations. Nous distinguons trois familles d'associations :

- un-à-plusieurs ;
- plusieurs-à-plusieurs ou classes-associations, et n-aires ;
- un-à-un.

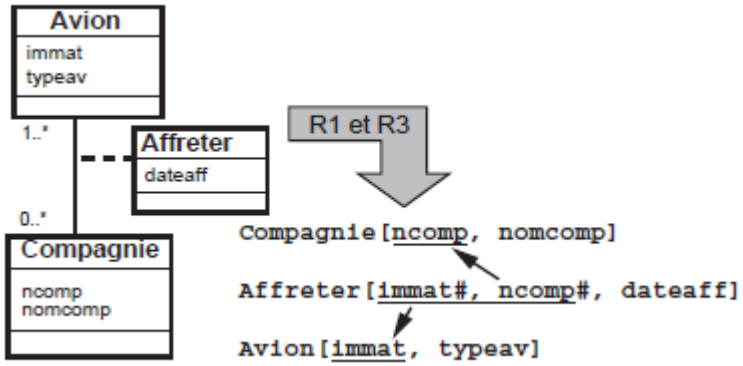
Associations un-à-plusieurs

R2 Il faut ajouter un attribut de type clé étrangère dans la relation Avion. L'attribut porte le nom de la clé primaire de la relation Compagnie.

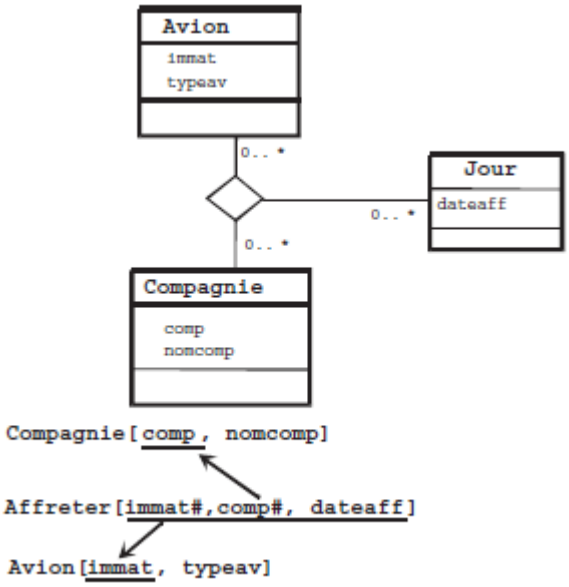


Associations plusieurs-à-plusieurs et n-aires

R3 La classe-association devient une relation dont la clé primaire est composée par la concaténation des identifiants des classes connectés à l'association. Les attributs de la classe-association doivent être ajoutés à la nouvelle relation.



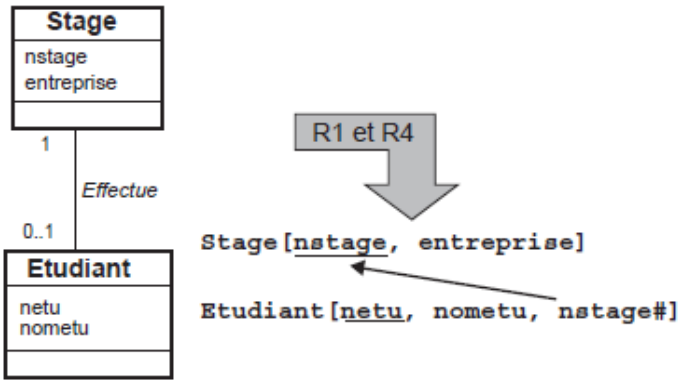
Transformation d'une association plusieurs-à-plusieurs



Transformation d'une association n-aire

Associations un-à-un

R4 La règle est la suivante, elle permet d'éviter les valeurs NULL dans la base de données.
Il faut ajouter un attribut clé étrangère dans la relation dérivée de la classe ayant la multiplicité minimale égale à un. L'attribut porte le nom de la clé primaire de la relation dérivée de la classe connectée à l'association.
Si les deux cardinalités (multiplicités) minimales sont à zéro, le choix est donné entre les deux relations dérivées de la règle R1. Si les deux cardinalités minimales sont à un, il est sans doute préférable de fusionner les deux classes en une seule.



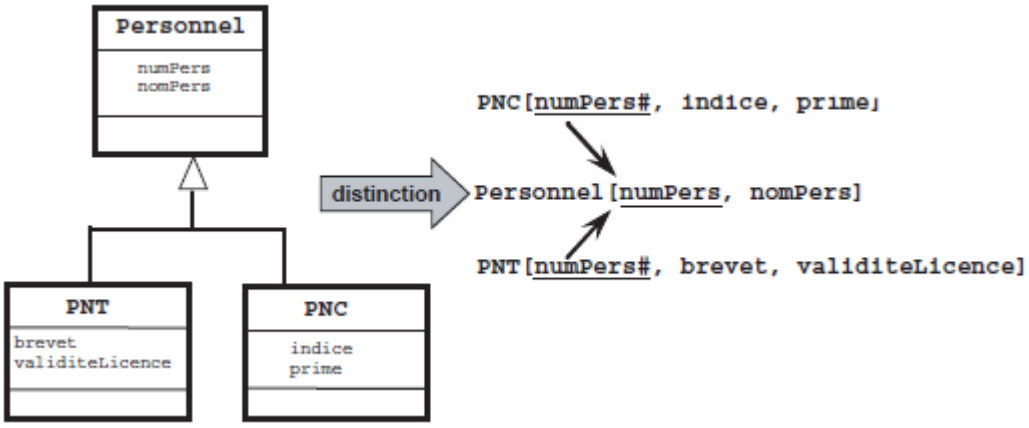
Transformation de l'héritage

Trois décompositions sont possibles pour traduire une relation d'héritage en fonction des contraintes existantes :

- décomposition par distinction ;
- décomposition descendante (push-down). S'il existe une contrainte de totalité ou de partition sur l'association d'héritage, il y aura deux cas possibles de décomposition ;
- décomposition ascendante (push-up).

Décomposition par distinction

Il faut transformer chaque sous-classe en une relation. La clé primaire de la sur-classe migre dans la (les) relation(s) issue(s) de la (des) sous-classe(s) et devient à la fois clé primaire et clé étrangère.

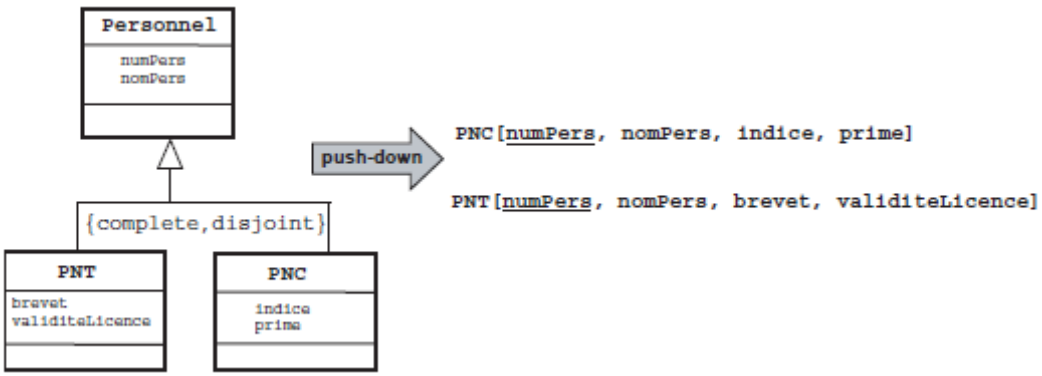


Décomposition par distinction d'une association d'héritage

Décomposition descendante (push-down)

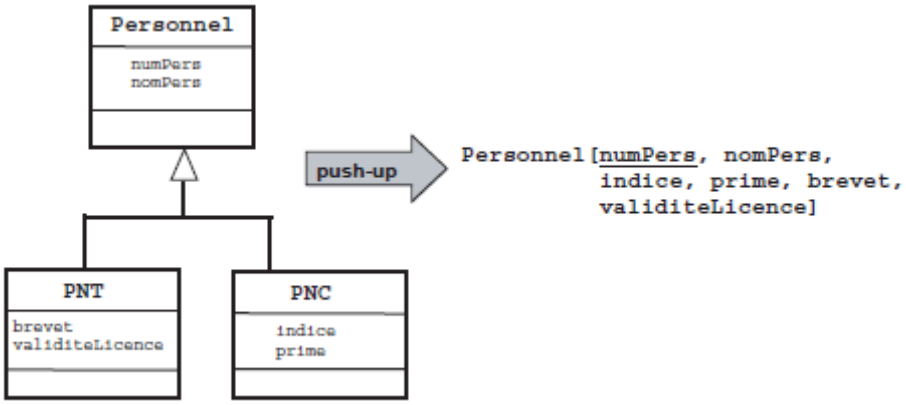
Deux cas sont possibles :

- S'il existe une contrainte de totalité ou de partition sur l'association, il est possible de ne pas traduire la relation issue de la sur-classe. Il faut alors faire migrer tous ses attributs dans la (les) relation(s) issue(s) de la (des) sous-classe(s).
- Dans le cas contraire, il faut faire migrer tous ses attributs dans la ou les relation(s) issue(s) de la (des) sous-classe(s) dans la (les) relation(s) issue(s) de la (des) sous-classe(s).



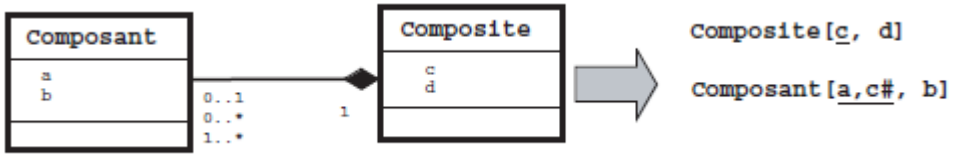
Décomposition ascendante (push-up)

Il faut supprimer la (les) relation(s) issue(s) de la (des) sous-classe(s) et faire migrer les attributs dans la relation issue de la sur-classe.



Transformation d'une composition

Alors que l'agrégation partagée de UML se traduit au niveau logique comme une simple association, il n'en est pas de même pour la composition.



Transformation d'une composition

5 Références

[1] Pascal ROQUES, Franck VALLÉE « UML en Action De l'analyse des besoins à la conception en Java ». Edition Eyrolles (2ème édition)

[2] Christian SOUTOU « UML2 pour les bases de données » Edition Eyrolles