1 Le modèle relationnel

1.1 Système d'emprunt dans une bibliothèque

On souhaite mettre en place un système informatique permettant la gestion des emprunts de livres d'une bibliothèque via des bornes automatiques. Dans ce système, on veut pouvoir stocker un certain nombre de données concernant :

— les *usagers* ;

— les *livres*.

De plus, ce système doit proposer de garder en mémoire les *emprunts* réalisés.

Un tel système, décrivant un certain nombre d'objets (usagers, livres) et de processus (emprunts) est appelé un **système d'information** (ici réalisé de manière informatique).

Il est important de noter que les données stockées par le système forment une approximation de la réalité, puisque ne sont retenues que les données pertinentes au regard du service. Par exemple, on peut retenir le nom, le prénom, la date de naissance d'un usager mais la couleur de ses yeux n'est pas pertinente. On dit qu'on fait une **modélisation** de la réalité.

Précisons un peu plus les spécifications de notre système :

En plus de retenir des informations, notre système doit permettre aux documentalistes d'**ajouter ou retirer** un usager ou livre du système.

Ces ajouts et retraits doivent pouvoir s'effectuer **sans erreur** : par exemple, il ne faut pas inscrire le même usager deux fois ou pire, retirer du système un usager ayant encore un emprunt en cours.

De plus, on doit garantir des **niveaux d'accès** aux données différents en fonction de l'utilisateur du système : les documentalistes peuvent accéder à toutes les informations et fonctionnalités alors que les usagers ne peuvent que déclarer un emprunt via les bornes.

Enfin, chaque borne devant accéder ou modifier des données communes, il est ici primordial de centraliser les données sur un serveur dont les bornes sont les **clients**.

Les caractéristiques listées ci-dessus sont des caractéristiques d'un **système de gestion de base de données (SGBD)**. Il s'agît d'un logiciel permettant la mise en oeuvre informatique d'un système d'informations.

Nous allons étudier les SGBD au fil des chapitres suivants mais pour l'heure, nous allons nous familiariser avec la manière dont ceux-ci organisent les données : **le modèle relationnel**.

1.2 Entités, relations, schémas

Toutes les informations stockées dans notre système peuvent être rangées dans trois ensembles : *Usagers*, *Livres* et *Emprunts* appelés des **relations**.

Voici quelques exemples d'objets (ou plutôt ici **entités**) pouvant être stockées dans ces relations :

— Une usagère : Amantine Dupin, née le 01/07/1994, demeurant 15 rue Meslay, à Paris 75003 ;

— Un livre : Notre Dame de Paris, par Victor Hugo, édité par Librairie Générale Française, ISBN :978-2-253-00968-9, 2010 ;

— Un emprunt : emprunt de Notre Dame de Paris par Amantine le 05/10/2020.

Une **base de donnée** est un ensemble de relations.

On a trois types d'entités : usager, livre et emprunt. Une entité est définie par plusieurs **attributs** pour lesquels on précise un type informatique (on parle ici de **domaine**) :

— Livre :
  — titre `String`
  — auteur `String`
  — éditeur `String`
  — isbn `String`
  — année d'édition `Int`

— Usager :
  — prénom `String`
  — nom `String`
  — date de naissance `Date`
  — adresse `String`

— Emprunt :
  — usager emprunteur
  — livre emprunté
  — date de l'emprunt `Date`

Résumons :

— une **base de donnée** est un ensemble de relations ;

— une **relation** est un ensemble d'entités ;

— une **entité** est un n-uplet d'attributs ;

— un attribut est une information stockée sous un certain type informatique, son domaine.

Un exemple de BD :

```
Livres :
        Notre Dame de Paris, Victor Hugo, Librairie Générale Française, 978-2-253-00968-9, 2010
        Le huitieme sortilège, Terry Pratchett, Pocket, 978-2-266-21182-6, 1993
        Logicomix, Apostolos Doxiadis, Vuibert, 978-2-311-10232-1, 2018

Usagers :
        Amantine Dupin, 01/07/1994, 15 rue Meslay Paris, 75003
        Clark Kent, 01/01/1986, 1 rue de la Mer Saint Genis Pouilly, 01630

Emprunts :
        Amantine, Dupin, Notre Dame de Paris, 05/10/2020
        Clark, Kent, Le huitième sortilège, 02/10/2020
        Clark, Kent, Logicomix, 02/10/2020
```

Chaque relation d'une base de donnée se conforme à un **schéma** décrivant le nom et le domaine de chacun de ses attributs. Lorsqu'on conçoit une base de donnée, la description de son schéma (l'ensemble des schémas de ses relations) est une information primordiale.

On peut décrire un schéma sous diverses formes (on l'a déjà fait précédemment sous forme de listes). La forme qu'on utilisera ici sera la suivante :

Relation(*nom1* `Domaine1`, *nom2* `Domaine2`, etc.)

Par exemple, le schéma de notre relation *Usagers* sera noté :

Usagers(*prénom* `String`, *nom* `String`, *date_ naissance* `Date`, *adresse* `String`)

2 Modélisation relationnelle des données, contraintes d'intégrité

2.1 Principes généraux

Lorsque l'on conçoit un base de donnée, la modélisation des données suit les étapes suivantes :

1. on détermine les entités (objets, personnes, processus, etc.) représentés dans la base ;
2. on rassemble ces entités dans des relations dont on donne les schémas en s'attachant à bien choir les attributs et leurs domaines ;
3. on définit les **contraintes d'intégrité** de la base de donnée, c'est à dire l'ensemble des règles logiques devant être respectées à tout moment par les données afin d'en garantir le bon fonctionnement.

Il existe différents types de contraintes d'intégrité : les contraintes de domaine, d'entité, de référence et les contraintes utilisateur.

Ces contraintes rendent certaines réalisation de relations valides (respectant toutes les contraintes spécifiées) ou invalides, un SGBD n'acceptant que des relations valides.

2.2 Contraintes de domaine

Les contraintes de domaines sont définies dans le schéma comme on l'a déjà vu.

Prenons par exemple la relation R(*a* `Int`, *b* `Boolean`) :

— R={(1, 42) , (2, `True`)} est une relation invalide car l'attribut *b* de la première entité n'est pas un booléen ;

— R={(1, `False`) , (2, `True`)} est une relation valide car les attributs *a* des deux entités sont bien des entiers et les *b* des booléens.

Le second exemple sera donc accepté par un SGBD mais pas le premier.

On a déjà pris en compte les domaines dans la première description de notre exemple de bibliothèque. Cependant, certains points restent insatisfaisants comme le format dans lequel on stocke l'adresse de l'usager. Pour rendre celui-ci plus pratique à manipuler, il pourrait être utile de décomposer l'attribut *adresse* en trois attributs : *adresse*, *code_postal* et *ville* tout trois de type `String` :

Usagers(*prénom* `String`, *nom* `String`, *date_naissance* `Date`, *adresse* `String`, *cp* `String`, *ville* `String`)

Remarque 1 : Il est ici important de stocker le code postal comme une chaîne de caractère et non comme un entier pour pouvoir afficher les zéros en début de code.

Remarque 2 : En revanche, ces contraintes n'empêche absolument pas de rentrer des valeur aberrantes. Par exemple, entrer `"toto"` pour l'attribut *pc* n'est ici pas interdit par la BD.

Les noms des différents domaines dépendent du SGDB utilisé en pratique, on donne donc ici des noms génériques qui seront éventuellement à décliner en fonction du logiciel choisi :

- `String` pour les chaînes de caractères ;
- `Int` pour les entiers ;
- `Boolean` pour les booléens ;
- `Float` pour les flottants ;
- `Date` pour les dates (jour/mois/année) ;
- `Time` pour les instants (heure : minute : seconde).

2.3 Contraintes d'entité

Imaginons dans notre exemple n'avoir stocké dans la relation *Usagers* uniquement les attributs *nom* et *prénom*. Cela aurait pu poser un problème car il est courant d'avoir plusieurs personnes ayant le même nom et le même prénom (amusez vous à chercher vos homonymes sur internet).

Pour palier à ce problème, on ajoute dans les schémas une information supplémentaire : la **clé primaire** de la relation. Une clé primaire est un attribut ou un ensemble d'attributs unique à chaque entité d'une relation donnée. Pour notre représentation des schémas, on soulignera d'un trait plein les clés primaires.

Souvent, afin d'être certain d'avoir une clé primaire, on ajoute un numéro d'identification unique à chaque entité. Par exemple, on peut rajouter à notre relation *Usagers* un code barre associé à sa carte de bibliothèque qui sera unique et servira de clé primaire :

Usagers(*code* Int, *prénom* `String`, *nom* `String`, *date_naissance* `Date`, *adresse* `String`, *cp* `String`, *ville* `String`)

Comme précisé dans la définition, une clé primaire d'une relation peut être constituée de plusieurs attributs, on parle alors de **clé primaire composite**. Cela signifie qu'il ne peut pas y avoir deux entités dans la relation partageant exactement les mêmes valeurs sur les deux attributs.

Prenons l'exemple de la relation R(*a* `Int`, *b* `Int`, *c* `Int`). Ici on a une clé primaire composite composée des attributs a et b et dans ce cas :

- R={(1,2,3) , (1,2,4)} est une relation invalide ;
- R={(1,2,3) , (1,3,4)} ou R={(2,2,3) , (1,2,4)} sont valides.

Pour revenir sur notre exemple principal, il eût été possible de déclarer comme clé primaire composite l'ensemble des attributs *nom*, *prénom*, *date_naissance* et *adresse* en supposant raisonnablement qu'il n'existe pas deux homonymes nés le même jour et demeurant à la même adresse. Cependant la solution du numéro d'identification est à la fois plus sûre, plus simple et, de fait, plus utilisée en pratique.

2.4 Contraintes de référence

Un autre problème dans notre exemple : rien n'interdit a priori d'entrer dans la relation *Emprunts* un emprunt réalisé par un usager ou d'un livre non-référencé dans les relations *Livres* ou *Usagers*.

Pour palier à ce problème, on rajoute une nouvelle information à nos schémas : les **clés étrangères**. Une clé étrangère est un attribut devant faire référence à un attribut existant dans une autre relation (d'où le "étrangère"). Dans notre représentation des schémas, on soulignera d'un trait en pointillés les clés secondaires et on précisera à quelle relation voire à quel attribut de relation la clé étrangère doit faire référence.

Il est assez fréquent d'utiliser comme clé secondaire d'une relation une clé primaire de la relation référencée.

Attention : un attribut déclaré comme clé secondaire dans une relation force cet attribut à être présent dans la relation étrangère s'il est présent dans la relation courante.

Par exemple pour *R1*(*a* `Int`, *b* `Int`), *R2*(*a* `Int`, *c* `Boolean`) :

— *R1*={(1,12), (2,70)}, *R2*={(1,True)} est valide car le $a = 1$ de $R2$ est bien référencé dans $R1$;

— *R1*={(1,2)}, *R2*={(2,True)} est invalide car il n'existe aucune entité de $R1$ d'attribut $a = 2$ comme suggéré dans $R2$.

On peut maintenant écrire le schéma final de notre base de donnée en intégrant le *code* utilisateur et un *num* (numéro) servant de clés primaires aux relations *Usagers* et *Livres* et de clés secondaires à la relation *Emprunts* :

— *Usagers*(*code* `Int`, *prénom* `String`, *nom* `String`, *date_naissance* `Date`, *adresse* `String`, *cp* `String`, *ville* `String`) ;

— *Livres*(*num* `Int`, *titre* `String`, *auteur* `String`, *isbn* `String`, *année* `Int`) ;

— *Emprunts*(*code* `Int`, *num* `Int`, *date* `Date`) où *code* fait référence au *code* de *Usagers* et *num* fait référence au *num* de *Livres*.

Remarque : on a de plus déclaré *code* comme clé primaire de *Emprunts* de sorte que pour un livre donné, un seul emprunt soit possible.

2.5 Contraintes utilisateur

On termine avec les contraintes utilisateurs qui sont souvent des contraintes de format où de plage de valeurs n'étant pas exprimables dans notre formalisme des schémas.

Par exemple, on pourrait demander à ce qu'une adresse mail contienne un et un seul symbole `@`, qu'un code postal comporte exactement 5 caractères tous étant des chiffres ou encore qu'une date de naissance ne soit pas aberrante (ex : personne née avant 1900).

Certaines de ces contraintes peuvent être prises en compte par les SGBD. Pour l'instant, on se contentera de les préciser en français à la suite de nos schémas.

Voici donc la forme finale de notre schéma (en faisant un peu de zèle) :

— *Usagers*(*code* `Int`, *prénom* `String`, *nom* `String`, *date_naissance* `Date`, *adresse* `String`, *cp* `String`, *ville* `String`) ;

CU : uniquement des caractères simples dans *prémon* et *nom*, *ville* existante, *cp* de cinq caractères numériques ;

— *Livres*(*num* `Int`, *titre* `String`, *auteur* `String`, *isbn* `String`, *année* `Int`) ;

CU : uniquement des caractères simples dans *auteur*, *isbn* au bon format ;

— *Emprunts*(*code* `Int`, *num* `Int`, *date* `Date`) où *code* fait référence au *code* de *Usagers* et *num* fait référence au *num* de *Livres*.

CU : *date* antérieure à la date courante.