Bases de Données

C2 - Conception et Modélisation

Lina Soualmia

Université de Rouen
LITIS - Équipe TIBS-CISMeF
lina.soualmia@chu-rouen.fr

9 septembre 2016

- 1. Étape conceptuelle : Conception et Modélisation de bases de données
  Utilisation de :
  - Méthodes, Modèles, Formalismes
  - Modèle Entité-Association E/A, Modèle Entité-Association étendu
  - Modèles Objet, Formalisme UML
    Power AMC, Power Designer WinDev, Oracle Designer Rational Rose, ...
- 2. Étape logique : Implantation d'une base de données
  - Modèle Relationnel / Modèle Objet-Relationnel / Modèle Objet
  - Optimisation du schéma (Normalisation, Dé-normalisation ...)

Conception et Modélisation

- 3. Étape physique :
  - SGBD Relationnel / SGBD Objet-Relationnel / SGBD Orienté Objet
  - Langages (SQL, PL/SQL, PRO*C, JDBC, Java, ...)
  - Optimisations (Groupement, Index, ...)
  - Administration
    Oracle, DB2, MySQL
- 4. Logiciels (SGBD, Interfaces, ...) & Matériels

Conception du schéma des bases

- C'est une des tâches essentielles des développeurs de bases de données
- Objectif : structuration du domaine d'application afin
  - de le représenter sous forme de types et de tables
  - d'accompagner ces structures de contraintes sur les données et d'en tirer plus de sémantique

La représentation

Elle doit être :
- **juste** pour éviter les aberrations sémantiques, notamment dans les résultats des requêtes ;
- **complète** pour permettre le développement des programmes d'application souhaitées ;
- **évolutive** pour supporter la prise en compte rapide de nouvelles demandes.

La démarche

Démarche de conception traditionnelle :
- par abstractions successives
- en descendant depuis les problèmes de l'utilisateur vers le Système de Gestion de Bases de Données.

Cinq étapes :

1. Perception du monde réel et capture des besoins
2. Élaboration du schéma conceptuel
3. Conception du schéma logique
4. A□nement du schéma logique
5. Élaboration du schéma physique

Remarque

- Étape 1 : plutôt relative au domaine du génie logiciel
- Étapes 2, 3, 4 et 5 : relatives au domaine des bases de données

Étape 1 : Perception du monde réel et capture des besoins

- Étude des problèmes des utilisateurs
- Compréhension de leurs besoins
  −→ Mise en place d'entretiens, d'analyses des flux d'information et des processus métier

Di□culté : Compréhension du problème dans son ensemble
−→ Réalisation des études de cas partiels par les concepteurs
Résultat : ensemble de vues ou schémas externes devant être intégrés dans l'étape suivante
Vues exprimées dans un modèle de de données : de type éntité-association ou objet, selon la méthode choisie.

Étape 2 : Élaboration du schéma conceptuel

- Intégration des schémas externes obtenus a l'étape précédente
- Chaque composant est un schéma conceptuel : diagramme entité-association ou diagramme de classes
- Résultat : modèle de problème représentant une partie de l'application
- Di□culté : intégration de toutes les parties dans un schéma conceptuel global complet, non redondant et cohérent

NB : des allers et retours avec l'étape précédente sont souvent nécessaires.

Étape 3 : Conception du schéma logique

Transformation du schéma conceptuel en structures de données supportées par le système choisi : le schéma logique.

- Avec un SGBD relationnel : passage à des tables.
- Avec un SGBD relationnel-objet : génération de types et de tables. (les types sont réutilisables)
- Avec un SGBD objet : génération de classes et d'associations

NB : Cette étape peut être complètement automatisée.

Étape 4 : A□nement du schéma logique

- Vérification : le schéma logique est-il un "bon" schéma ?
- Définition en première approximation : *un bon schéma est un schéma sans oublis ni redondances d'informations*
- Plus précisément : *un schéma est bon si le modèle relationnel associé respecte au moins la troisième forme normale* et la forme normale de Boyce-Codd (à voir plus loin)
- Objectif en relationnel : regrouper ou décomposer les tables de manière a représenter fidèlement le monde réel modélisé

Étape 5 : Élaboration du schéma physique

- Prise en compte de toutes les transactions concernant les applications traitées
- Permet de déterminer les accès fréquents
- Choix des bonnes structures physiques :
  - groupement ou partitionnement de tables
  - choix des index, etc.
- C'est le point essentiel pour obtenir de bonnes performances.

Schéma Co ceptuel

- Modélisation du problème en utilisant les spécifications des besoins obtenues a l'étape 1 (capture des besoins)
- Deux possibilités :
  - utilisation du formalisme Entité Relation (ou Entité Association)
    $\longrightarrow$ production d'un diagramme ER/EA
  - utilisation du formalisme UML
    $\longrightarrow$ production d'un digramme de classes

Indépendance du modèle conceptuel par rapport au schéma physique

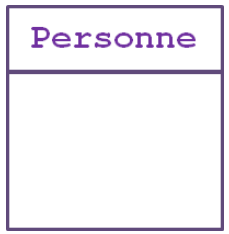
Phases d'élaboration du schéma conceptuel

- Identification des entités ou classes
- Identification des associations
- Identification des attributs pour chacune des entités ou classes
- Définition des identifiants

Identification des entités ou classes

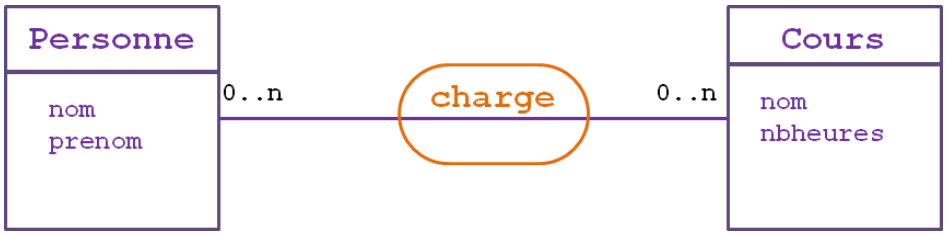
- Entités : élément abstrait ou concret (objet, évènement, etc.) reconnu distinctement
  Exemples : Jean Dupont, Michel Durant
- Type-entités : ensemble des entités ayant les mêmes caractéristiques
  Ex. : `Personne (nom, prenom)`
- Par abus de langage, on parle souvent d'entités à la place de type-entités

Dans l'étape 1, il s'agit de la description des éléments.



Identification des associations

- Association : Lien logique entre deux entités
- Type-Association : Ensemble d'associations ou de relations possédant les mêmes caractéristiques.
- Association/type-association : même abus de langage
- À l'étape 1 : une phrase simple reliant deux entités
  Ex. : un professeur est en charge de cours (lien entre les entités `professeur` et `cours`)
- Plusieurs types d'association existent



Types d'associations

- **u aire** : relation au sein d'une même entité
  Ex. : un employé supervise un employé
- **bi aire** : relation entre deux entités (di□érentes)
  Ex. : un client passe plusieurs commandes
- **ter aire** : relation entre trois entités (di□érentes)
  Ex. : un internaute note un film à différentes dates (on veut conserver l'historique des notes).

Cardinalité d'un type-association

- Cardinalité : nombre minimal et maximal de fois qu'une entité peut intervenir dans une association de ce type
  Ex. : un client peut commander 1 à n produits
- Remarques :
  - la cardinalité minimale doit être inférieure à la cardinalité maximale
  - la cardinalité doit être associée à chaque patte de la relation

Cardinalité minimale/maximale

- Cardinalité minimale :
  - 0 : une entité peut exister tout en n'étant impliquée dans aucune association
  - 1 : une entité ne peut exister que si elle est impliquée dans au moins une association
  - n : une entité ne peut exister que si elle est impliquée dans plusieurs associations (cas rare, à éviter car cela pose des problèmes)
- Cardinalité maximale :
  - 0 : une entité ne peut pas être impliquée dans une association (normalement inexistant sinon problème de conception)
  - 1 : une entité peut être impliquée dans au maximum une association
  - n : une entité peut être impliquée dans plusieurs associations

Identification des attributs

- Attribut : caractéristique associée a une entité
  Ex. : *nom, prenom, age*
- Domaine associé à un attribut : ensemble des valeurs possibles
- Chaque attribut doit posséder une valeur compatible avec son domaine
- Remarque : Éviter absolument les attributs calculés. Toujours utiliser des données primaires - les attributs qui servent à les calculer.

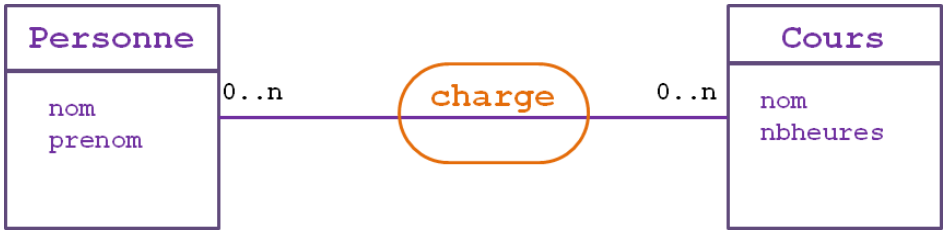
Définition de l'identifiant

- Identifiant : liste des attributs devant avoir une valeur unique dans chaque entité
  Ex. : *numéro d'immatriculation d'une voiture, numéro de sécurité sociale*
- Remarques :
  - On utilise plutôt le terme **clé** que identifiant
  - Chaque type doit posséder un identifiant (formé d'un ou de plusieurs attributs)
  - L'identifiant d'une association est la concaténation des identifiants des entités liées
    NB : on peut définir un identifiant plus naturel

Remarques sur la conception

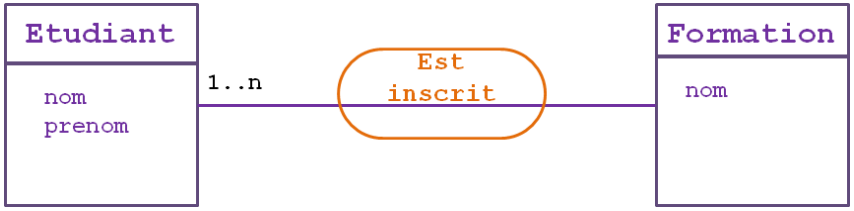
- Un attribut ne peut être partagé entre deux entités ou associations
  - problème de redondance
- En cas de difficulté à choisir entre entité et association :
  - utiliser le contexte pour y répondre
- En cas de difficulté à trouver un identifiant pour un type-entité :
  - ne s'agirait-il pas d'une association ?
- Association dont toutes les pattes ont une cardinalité 1,1 :
  - l'association et les entités liées ne correspondraient-ils pas a une seule entité ?

- Formalisme ER :



- Formalisme UML :

(une des cardinalités est volontairement absente)



Tout `étudiant` participe au moins une fois à l'association `est inscrit`.

Tout étudiant est inscrit dans au moins une formation

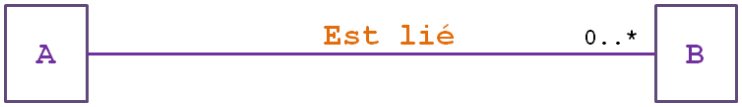
Autrement dit : une instance d'étudiant peut être associée à plusieurs formations.

Formalisme ER :

Interprétations :

- A est lié 0 à n fois à B
- La connaissance de B permet de définir A
- La clé de B définit l'instance de A

Formalisme UML :



ER ou UML ?

- Si conception de bases de données :
  - utilisation du modèle entité/relation
  - On met l'accent sur le système d'information (stockage, traitement, réception, diffusion de l'information)
- Si conception objet et programmation :
  - utilisation de UML (2 incluant l'héritage).
  - On met l'accent sur les structures de données et la programmation

Élaboration du schéma logique

Transformation du modèle conceptuel en une structure de données basée sur un modèle de données spécifique (par exemple relationnel)

- Réalisation de la transformation a l'aide de règles formelles
  −→ Possibilité d'automatisation de cette étape (Objecteering, Rational Rose)
- Indépendant de la couche physique
- Résultat : modèle logique de la base de données

Passage au relationnel

- Implémentation des entités et associations sous forme de tables
- Les attributs correspondent aux colonnes des tables
  - le nom de l'attribut est le nom de la colonne
  - l'ensemble des valeurs possibles est le domaine
- Exemple :
  - *Professeur(numProf, nom, prenom)*
  - *Cours(nomCours, nom, nbheures)*
  - *Charge(numProf, numCours)*

Principe

- Traduction des associations :
  Règle de base : représentation des associations par une table dont
  - le schéma est le nom de l'association
  - la liste des clés des entités participantes suivie des attributs de l'association
- Amélioration :
  - Regrouper les associations 1..n avec la classe cible
- Exemple :
  - *Voiture(numV, Marque, modele)*
  - *Possede(numProp, numV, Date)*
    $\longrightarrow$ les deux tables peuvent être regroupées si toutes les voitures n'ont qu'un et un seul propriétaire

Formes ormales

Respecter les formes normales

Pourquoi normaliser ?

- pour limiter les redondances des données
- pour limiter les pertes de données
- pour limiter les incohérences au sein des données
- pour améliorer les performances des traitements

8 formes normales :

- Formes normales 1 à 3
- Forme normale de Boyce-Codd
- Formes normales 4/5(/6)
- Forme normale de domaine-clé

Objectifs des trois premières formes normales : permettre la décomposition de relations sans perte d'informations

Une relation en forme normale de niveau N est forcément de forme normale de niveau N - 1

Première forme normale (1FN)

Une relation est en première forme normale si tous ses attributs contiennent des valeurs

- simples et non-décomposables (utiliser une liste ou une table externe)
- non-répétitives
- constantes dans le temps (date de naissance plutôt que l'âge)

Exemple

`Vol(NoVol, CodeAeroDep, CodeAeroArr, HeureDep, HeureArr, Jours)`

devient

`Vol(NoVol, CodeAeroDep, CodeAeroArr, HeureDep, HeureArr)`

`Vol(NoVol, Jour)`

Deuxième forme normale (2FN)

Une relation est en deuxième forme normale si et seulement si :

- elle est en première forme normale
- tout attribut non clé est complètement dépendant de toute la clé

Autrement dit, une des trois conditions doit être respectée :

- La clé primaire n'est formée que d'un seul attribut
- La clé primaire contient tous les attributs de la table
- Si la clé a plus d'un attribut, une dépendance fonctionnelle ne doit jamais exister entre une partie seulement de la clé et un autre attribut de la table.

Exemple

`Avion(Constr, Modele, Conso, Capacite, VitesseMax)`
$\longrightarrow$ il y a une dépendance fonctionnelle entre *Modèle* et *Capacité*

On divise la table en deux :

`Avion(Constr, Modele)`

`ModeleAvion(Modele, Conso, Capacite, VitesseMax)`

Troisième forme normale (3FN)

Une relation est en troisième forme normale si et seulement si :

- elle est en deuxième forme normale
- tout attribut n'appartenant pas à une clé ne dépend pas d'un attribut non clé

$\longrightarrow$ les dépendances fonctionnelles entre deux attributs ordinaires (ne faisant par partie de la clé) ne sont pas autorisées

E emple

`Enseignant(Nom, Categorie, Classe, Salaire)`

$\longrightarrow$ Le salaire dépend de la *Catégorie* et de la *Classe*

devient

`Enseignant(Nom, Categorie, Classe)`

`Salaire(Categorie, Classe, Salaire)`

Forme normale de Boyce-Codd (BCNF)

- Extension plus rigide de la troisième forme normale (définie par R.F. Boyce et E.F. Codd - en partant du constat que la 3FN comportait certaines anomalies)
- Une relation est en forme normale de Boyce-Codd si et seulement si :
  - *aucun attribut faisant partie de la clé ne dépend d'un attribut ne faisant pas partie de la clé primaire*

Un modèle relationnel en FNBC est considéré comme étant de qualité suffisante pour une implantation
Les cas de relations en 3FN qui ne sont pas déjà en FNBC sont très rares

E emple

`R(A, B, C, D)`

Avec les dépendances : B, C $\rightarrow$ A ; B, C $\rightarrow$ D ; D $\rightarrow$ B, (ce qui entraîne de nombreuses redondances)

On propose les relations :

`R(A, B, C)`

`R'(D, B)`

ReservationCourtTennis(NomCourt, HeureDebut, HeureFin, ClasseTauxHoraire)

La classe du taux horaire (SILVER, GOLD, PREMIUM) détermine les courts disponibles.

On propose les relations :

ReservationCourtTennis(NomCourt, HeureDebut, HeureFin)

ClasseCourt(ClassTauxHoraire, NomCourt)

Objectifs :

- Rechercher de bonnes performances
- Prendre en compte les transactions
- Indexer, dénormaliser, grouper, partitionner les tables

Résultat : modèle physique optimisé de la base de données

Schéma relationnel :

COURS(NumCours, NomC, NbHeures, Annee)

PROFESSEURS(NumProf, NomP, Specialite, DateEntree, DerPromo, SalaireBase, SalaireActuel)

CHARGE(NumProf, NumCours)


```
create table  COURS
(NumCours NUMBER(2) NOT NULL,
  NomC  VARCHAR2 (20) NOT NULL,
  NbHeures NUMBER(2),
  Annee NUMBER(1),
  constraint  PKCours  primary key (NumCours));
create table  PROFESSEURS
(NumProf NUMBER(4) NOT NULL,
  NomP  VARCHAR2 (25) NOT NULL,
  Specialite  VARCHAR2 (20),
  DateEntree DATE,
  DerPromo DATE,
  SalaireBase NUMBER
  SalaireActuel NUMBER
  constraint  PKProfesseurs  primary key (NumProf));
```

```
create table  CHARGE
(NumProf  NUMBER(4) NOT NULL,
  NumCours NUMBER(4) NOT NULL,
  constraint PKCharge primary key (NumCours, NumProf));

alter table  CHARGE
  add constraint FKChargeCours foreign key (NumCours)
  references COURS(NumCours);

alter table  CHARGE
  add constraint FKChargeProfesseur foreign key  (NumProf)
  references PROFESSEURS (NumProf);
```



```
create table  COURS
(NumCours NUMBER(2),
  NomC VARCHAR2(20),
...
  constraint PKCours primary key (NumCours),
  constraint NNCoursNomC check (NomCours IS NOT NULL));

create table PROFESSEURS
(NumProf NUMBER(4),
  NomP VARCHAR2(25),
...
  constraint  PKProfesseurs primary key (NumProf),
  constraint  NNProfesseursNomP check (NomP IS NOT NULL));
```



```
create table  CHARGE
(NumProf  NUMBER(4),
  NumCours NUMBER(4),
  constraint PKCharge primary key (NumCours, NumProf));

alter table  CHARGE
  add constraint FKChargeCours foreign key (NumCours)
  references COURS(NumCours);

alter table  CHARGE
  add constraint FKChargeProfesseur foreign key  (NumProf)
  references PROFESSEURS (NumProf);
```