Bases de données

Objectifs du cours : étude des principes des Systèmes de Gestion de Bases de Données (SGBD) relationnels et la mise en pratique de ces principes.

Plan

1. Introduction
2. Conception d'un schéma relationnel
    - Bon et mauvais schéma
    - Le modèle Entité/Association
    - Le modèle relationnel
    - Passage d'un schéma E/A à un schéma relationnel
3. Langages d'interrogation et de manipulation
    - Algèbre relationnelle
    - SQL
    - Création de schémas relationnels
4. Pratique d'un SGBD relationnel
    - Introduction aux différentes architectures des SI
    - MySQL

Bibliographie

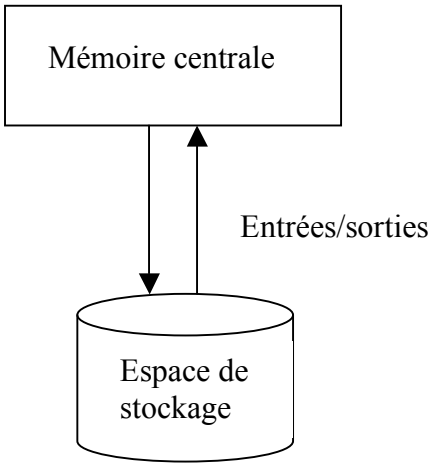
- Philippe Rigaux, Cour de bases de données, 2003, http://www.lamsade.dauphine.fr/rigaux/bd/?page=doc
- Paul Dubois, MySQL, CampusPress, 2000, http://www.mysql.com/

Introduction

Qu'est-ce qu'une donnée ?

Information quelconque, ex : *Voici une personne, elle s'appelle Nathalie.*
Relation entre des informations, ex : *Nathalie enseigne les bases de données.*
Des relations de ce genre définissent des *structures*.

Définition 1 : une base de données est un gros ensemble d'informations structurées mémorisées sur un support permanent.

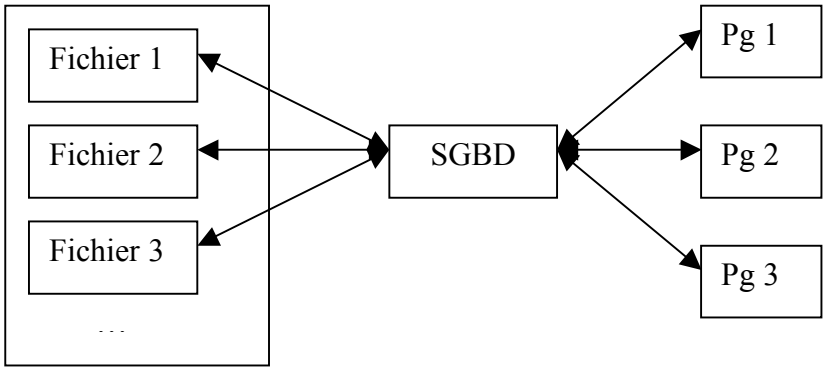


Une organisation consistant en un (ou plusieurs) fichier(s) stockés sur mémoire secondaire est conforme à cette définition. Malheureusement, l'utilisation directe de fichiers soulève de très gros problèmes :

- lourdeur d'accès aux données,
- manque de sécurité,
- pas de contrôle de concurrence d'accès.

→ Recours à un logiciel (un SGBD) chargé de gérer les fichiers constituant une base de données, de prendre en charge les fonctionnalités de protection et de sécurité et de fournir les différents types d'interface nécessaires à l'accès aux données.

Définition 2 : un Système de Gestion de Bases de Données (SGBD) est un logiciel de haut niveau qui permet de gérer les informations stockées dans une base de données, partageables entre plusieurs utilisateurs et/ou programmes et manipulées indépendamment de leur représentation physique.



Un SGBD est complexe à concevoir. On peut se raccrocher à une architecture standard conforme à la plus grande partie des SGBD existants. Elle distingue 3 niveaux

- Niveau physique : gestion sur mémoires secondaires des données, du schémas, des index ; partage des données et gestion de concurrence d'accès ; reprise sur pannes (fiabilité) ; distribution des données et interopérabilité (accès aux réseaux).
- Niveau logique : définition de la structure des données, langage de description de données (LDD), consultation et mise à jour des données, langage de requêtes (LR), langage de manipulation de données (LMD), gestion de la confidentialité (sécurité), maintien de l'intégrité.
- Niveau externe : vues, environnement de programmation, interface, outils d'aide, outils de saisie, d'impression d'états.

L'utilisation d'un SGBD suppose de comprendre les fonctionnalités suivantes :

1. Définition du schéma de données : description des données contenues dans la base conforme à un modèle de données qui propose des outils de description (structures, contraintes et opérations). Dans un SGBD, il existe plusieurs descriptions des mêmes objets, il y a
   - le modèle conceptuel qui décrit le système d'information,
   - le modèle logique qui fait l'interface avec le SGBD en proposant un LDD et un LMD indépendants de la représentation physique des données et de l'implantation des opérations, et des contraintes d'intégrité,
   - le modèle physique qui concerne l'accès aux fichiers.

   Quand on conçoit une BD, on décide la structure logique puis la structure physique, on écrit les programmes d'application en la structure logique et enfin le SGBD se charge de transcrire les commandes du LMD en instructions appropriées à la représentation physique.
2. Opérations sur les données : la création, la modification, la destruction et la recherche. Ce sont des commandes du LMD. La plus complexe est la recherche. Une bonne requête s'exprime facilement, doit être performante en temps de calcul et surtout doit être fiable. Le langage de requête le plus répandu est le SQL.
3. Partage des données entre plusieurs utilisateurs : éviter les blocages tout en empêchant des modifications anarchiques. Le SGBD doit savoir gérer les conflits de mise à jour, offrir un mécanisme de retour arrière si on décide d'annuler des modifications en cours et donner une image cohérente des données si l'un fait des requêtes et l'autre des mises à jour.
4. Optimisation des performances d'une requête en s'appuyant sur l'organisation physique des données : les fichiers séquentiels, les index et le regroupement des données par hachage. Un module particulier du SGBD, l'optimiseur, tient compte de cette organisation et des caractéristiques de la requête pour choisir la meilleure séquence des opérations.

Conception d'un schéma relationnel

1. Bons et mauvais schémas

FilmSimple

Titre	Annee	NomMES	PrenomMES	AnneeNaiss
Hana-bi	1997	Kitano	Takeshi	1947
Big fish	2003	Burton	Tim	1958
Edward aux mains d'argent	1990	Burton	Tim	1958
Sonatine	1993	Kitano	Takeshi	1947
Pulp Fiction	1995	Tarantino	Quentin	1963
Play Time	1967	Tati	Jacques	1907
Vertigo	1958	Hitchcock	Alfred	1899
Psychose	1960	Hitchcock	Alfred	1899
Parle avec elle	2002	Almodovar	Pedro	1949
Mon oncle	1958	Tati	Jacques	1907
Volver	2006	Almodovar	Pedro	1949
La mauvaise éducation	2004	Almodovar	Pedro	1949

Grave défaut : il est possible de représenter la même information plusieurs fois.

Anomalie d'insertion
Rien n'empêche de représenter plusieurs fois le même film avec des données différentes. Qu'est-ce qui distingue 2 films l'un de l'autre ? A quel moment peut-on dire que la même information est répétée ? Peut-il y avoir 2 films différents avec le même titre par exemple ?

Anomalie de modification
La redondance d'information entraîne des anomalies de mise à jour. Par exemple, la modification de l'année de naissance d'Hitchcock pour Vertigo et pas pour Psychose entraîne des informations incohérentes. Ne peut-on dire qu'il n'y a qu'un seul réalisateur nommé Hitchcock et qu'il ne doit y avoir qu'une seule année de naissance pour un réalisateur de ce nom ?

Anomalie de suppression
On ne peut pas supprimer un film sans supprimer son metteur en scène.

La bonne méthode
1. être capable de représenter individuellement les films et les réalisateurs, de manière à ce qu'une action sur l'un n'entraîne pas systématiquement une action sur l'autre,
2. définir une méthode d'identification d'un film ou d'un réalisateur, qui permette d'assurer que la même information est représentée une seule fois,
3. préserver le lien entre les films et les réalisateurs mais sans introduire de redondance.

Pour notre exemple, on distingue tout d'abord les tables *Films* et *Réalisateurs*. Ensuite, on décide que 2 films ne peuvent avoir le même titre, mais que 2 réalisateurs peuvent avoir les mêmes noms.

Films

Titre	Annee
Hana-bi	1997
Big fish	2003
Edward aux mains d'argent	1990
Sonatine	1993
Pulp Fiction	1995
Play Time	1967
Vertigo	1958
Psychose	1960
Parle avec elle	2002
Mon oncle	1958
Volver	2006
La mauvaise éducation	2004

Réalisateurs

idMES	NomMES	PrenomMES	AnneeNaiss
1	Kitano	Takeshi	1947
2	Burton	Tim	1958
3	Tarantino	Quentin	1963
4	Tati	Jacques	1907
5	Hitchcock	Alfred	1899
6	Almodovar	Pedro	1949

Il reste à représenter le lien entre les films et les réalisateurs sans introduire de redondance.

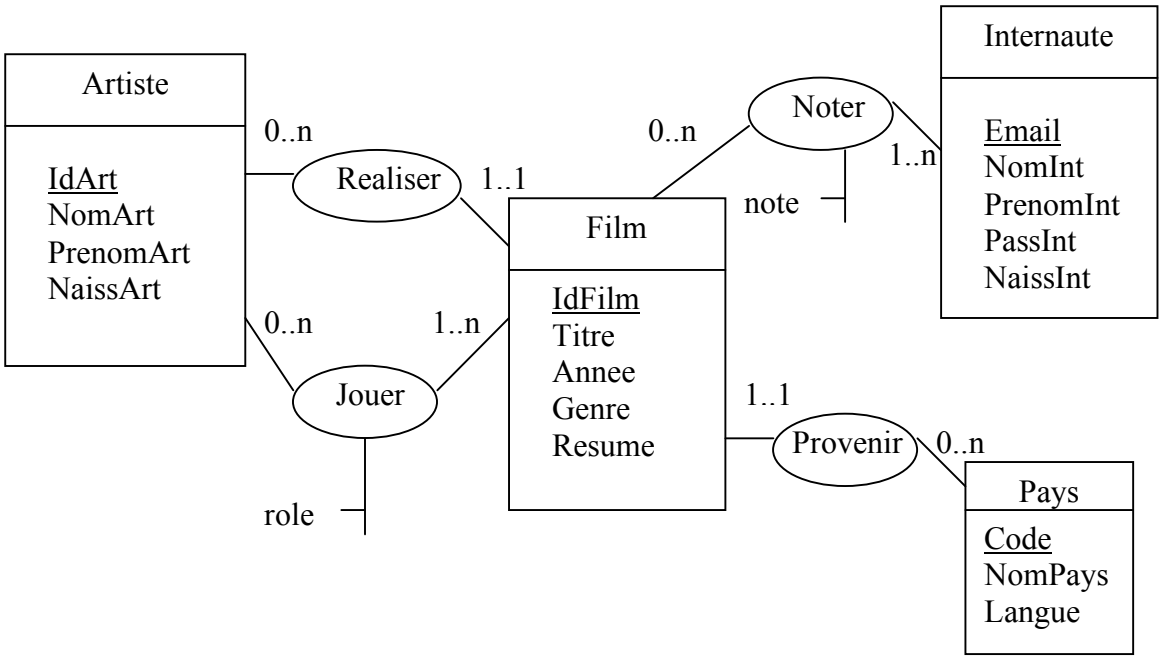
FilmsRéalisés

Titre	**IdMES**
Hana-bi	1
Big fish	2
Edward aux mains d'argent	2
Sonatine	1
Pulp Fiction	3
Play Time	4
Vertigo	5
Psychose	5
Parle avec elle	6
Mon oncle	4
Volver	6
La mauvaise éducation	6

Toutes les anomalies citées précédemment ont disparu.
La modélisation entité/association offre une méthode simple pour arriver au résultat ci-dessus et ce même dans des cas beaucoup plus complexes.

2. Le modèle Entité/Association

Utilisé presque universellement pour la conception des bases de données relationnelles. La conception d'un schéma correct est essentielle pour le développement d'une application viable. Conçu en 1976, il est à la base de la plupart des méthodes de conception. Il décrit l'application visée, cad une abstraction d'un domaine d'étude relativement aux objectifs fixés. Cela consiste à choisir certains aspects de la réalités perçue en fonction des besoins qui doivent être précisément définis.



Les entités

Définition 3 : on désigne par *entité* tout objet identifiable et pertinent pour l'application.

La notion d'*identité* est primordiale. C'est elle qui permet de distinguer les entités les unes des autres. On parle d'*identifiant* ou de *clé* comme moyen technique pour pouvoir effectuer cette distinction. La première étape d'une conception consiste à identifier les entités utiles. La deuxième est de regrouper les entités en ensembles.

Les attributs

Les entités sont caractérisées par des propriétés appelées attributs. On ne garde que les propriétés utiles pour l'application. Un attribut est désigné par un nom et prend ses valeurs dans un domaine énumérable. Un attribut prend une valeur et une seule, on dit qu'il est atomique.

Types d'entités

Elle exprime un type, une classe, un ensemble dont les éléments sont les entités (ou occurrences).

Définition 4 : Le type d'une entité est composé de son nom, de ses attributs et de son identifiant.

Une entité e est une instance du type E. Un ensemble d'entités $\{e_1, e_2, \ldots, e_n\}$ est une extension de E.

Définition 5 : soit E un type d'entité et A l'ensemble des attributs de E, une clé (ou identifiant) de E est un sous-ensemble minimal de A permettant d'identifier de manière unique une entité parmi n'importe quelle extension de E.

Il est possible d'avoir plusieurs clés pour un même ensemble d'entités. On en choisit alors une comme clé primaire et les autres comme clés secondaires. Le choix de la clé primaire est déterminant pour la qualité du schéma de la base.

Un identifiant doit être

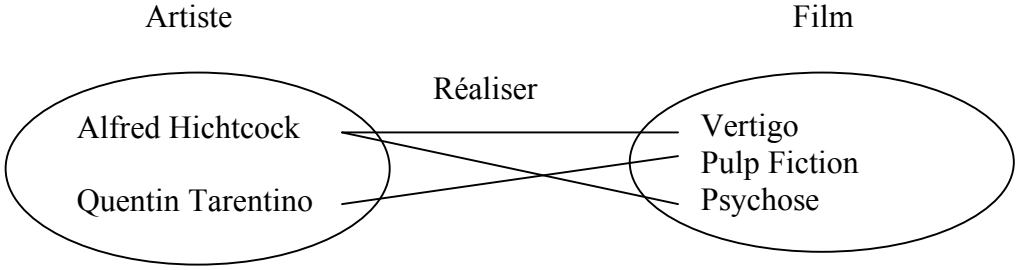
- *univalué* : à une occurrence correspond une seule valeur pour un identifiant donné,
- *discriminant* : à une valeur correspond une seule occurrence de l'individu
- *stable* : pour une occurrence donnée d'individu, une fois définie une valeur à son identifiant, cette valeur doit être conservée jusqu'à la destruction de l'occurrence
- *minimal* : s'agissant d'un identifiant composite, la suppression d'un de ses composants lui ferait perdre son caractère discriminant.

Dans la situation fréquente où on a du mal à déterminer quelle est la clé d'un type d'entité ; on crée un identifiant abstrait appelé id (un numéro séquentiel) indépendant de tous les autres attributs, de plus, sa taille de stockage est la plus petite possible.

Association binaire

Elle modélise un ensemble d'associations de même nature entre 2 occurrences d'individus (de types différents ou de même type).

Définition 6 : une association binaire entre les types d'entités E_1 et E_2 est un ensemble de couples (e_1, e_2) avec $e_1 \in E_1$ et $e_2 \in E_2$.



Définition 7 : soit une association (E_1, E_2) entre 2 types d'entités, la cardinalité de l'association pour E_i, $i \in \{1,2\}$ est une paire {min,max} où min (resp. max) désigne le nombre minimal (resp. maximal) de fois où une entité e_i de E_i peut intervenir dans l'association.

Les cardinalités n'expriment pas une vérité absolue mais des choix de conception.

Dans le cas d'associations avec des cardinalités multiples de chaque côté, on peut avoir des attributs qui ne peuvent être affectés qu'à l'association elle-même. Il s'agit d'informations qui ne peuvent prendre de sens qu'avec la présence des 2 types d'entités.

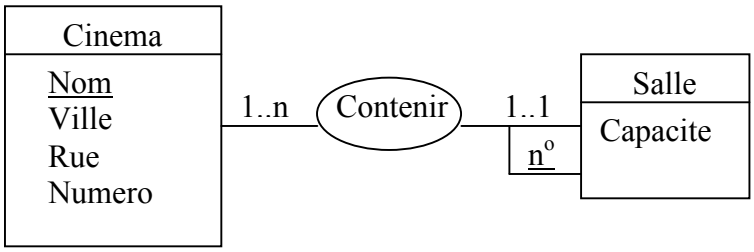
Remarque : toute association binaire avec cardinalité (1,1) ne peut être porteuse d'attribut sinon, elle migre obligatoirement dans l'entité portant cette cardinalité.

Définition 8 : la clé d'une association (binaire) entre un type d'entité E_1 et un type d'entité E_2 est le couple constitué de la clé c_1 de E_1 et de la clé c_2 de E_2.

Entité faible

Une entité ne peut exister qu'en étroite association avec une autre et est identifiée relativement à cette autre entité. On parle alors d'entité faible.

Exemple : un cinéma et ses salles.
On peut utiliser une association classique entre cinéma et salles, mais cela force à créer un identifiant id pour le type d'entité salle. Il est préférable de numéroter les salles par un numéro interne à chaque cinéma. La clé d'une salle est alors constituée de la clé du cinéma et du numéro de la salle. L'entité salle ne dispose pas d'une identification absolue mais d'une identification relative à une autre entité. Cela force la salle à toujours être associée à un et un seul cinéma.



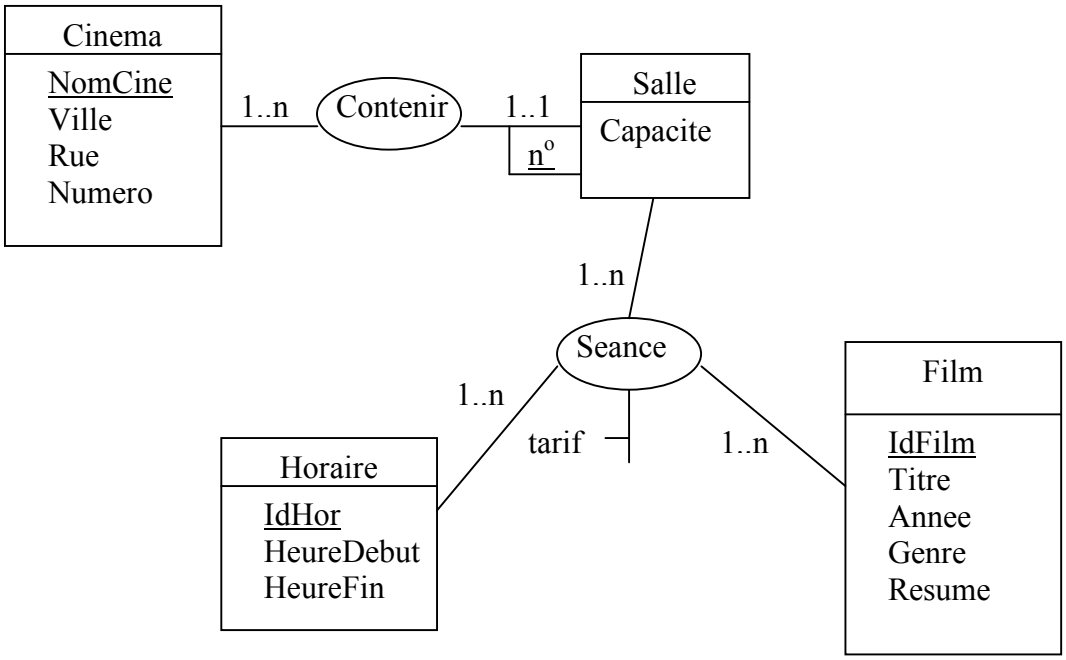
L'introduction d'entité faible n'est pas une nécessité absolue puisqu'on peut utiliser une association classique. Mais avec une entité faible on obtient un identifiant composé qui est souvent plus pratique à gérer et plus facile pour certaines requêtes.
La présence d'un type d'entité faible B associé à un type d'entité A implique aussi des contraintes fortes sur les créations, modifications et destructions des instances de B et A ; on doit s'assurer que la contrainte est valide.

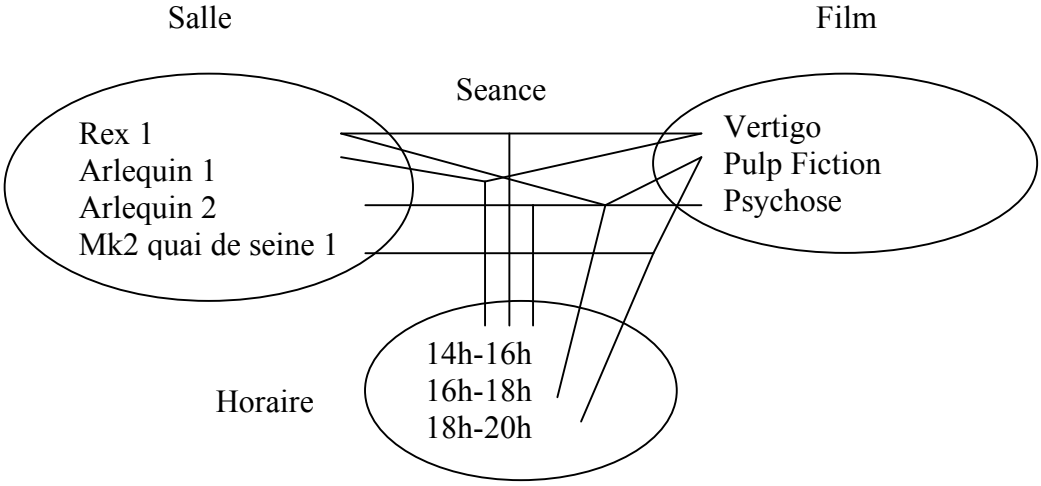
Associations généralisées

La définition d'une association n-aire est une généralisation de celle des associations binaires.

Définition 9 : une association n-aire entre n types d'entités E_1, E_2, …, E_n est un ensemble de n-uplets (e_1, e_2, …, e_n) où chaque e_i appartient à E_i.

On appelle *dimension* le nombre d'entités composant l'association. On appelle *collection* la liste de ces entités. L'occurrence d'une association est déterminée par les occurrences des entités de sa collection. La dimension d'une association est non limitée (en pratique maximum 4).



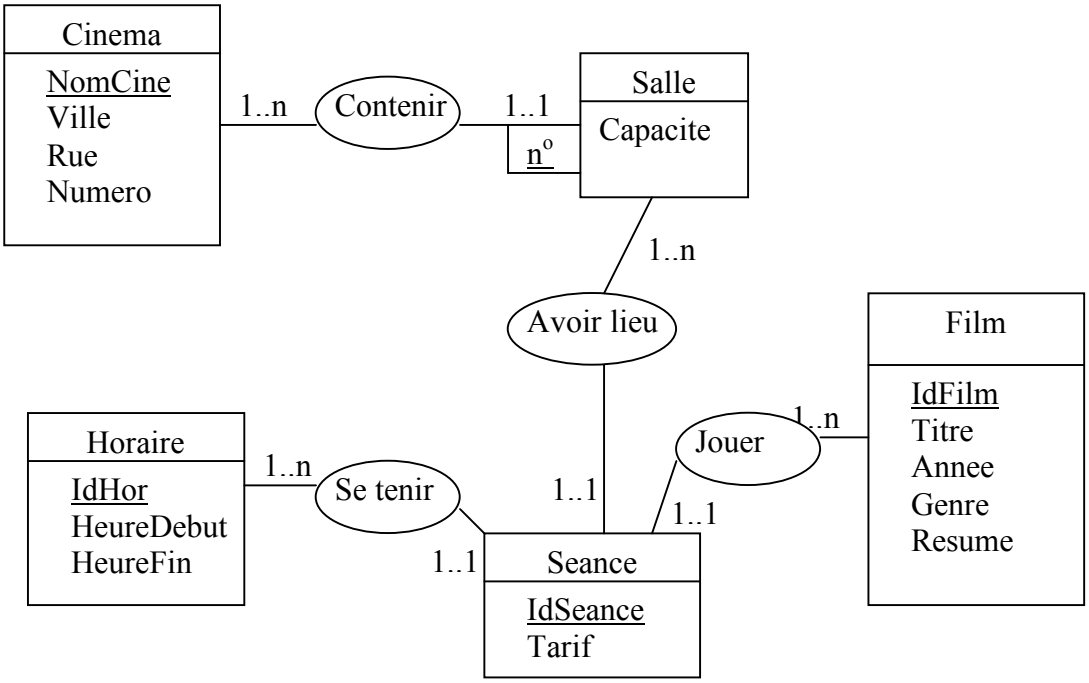


La détermination de la clé de l'association Seance est problématique, car elle doit être le n–uplet (NomCiné, noSalle, IdFilm, IdHor). Une telle clé est assez volumineuse et elle ne permet pas d'imposer certaines contraintes comme le fait que dans une salle, pour un horaire donné, il n'y a qu'un seul film.

Il est possible de remplacer l'association par un type d'identité. **Mais cela ne règle pas toujours le problème !!**

Règle : soit A une association entre les types d'entité (E_1, E_2, …, E_n), pour sa transformation en type d'entité, il faut

1. attribuer un identifiant autonome à A,
2. créer une association A_i entre A et chacun des E_i. de cardinalité 1..1 (du côté de A) et 1..n de l'autre.



Le problème de la salle persiste (dans une salle, pour un horaire donné, il est possible d'avoir plusieurs films).

Le modèle E/A permet de spécifier la structure des informations de la base et d'offrir une représentation abstraite indépendante du modèle logique.
En général, on essaie

1. d'éviter toute redondance,
2. de privilégier la simplicité et la lisibilité en ne représentant que ce qui est strictement nécessaire.

3. Le modèle relationnel

Le modèle relationnel (E. Codd, 1970) offre une totale indépendance entre les représentations logiques et physiques. Il n'existe qu'une seule structure la *relation* représentée par une *table*.

Films

Titre	Annee	Genre
Hana-bi	1997	Drame
Big fish	2003	Comedie dramatique
Edward aux mains d'argent	1990	Fantastique
Sonatine	1993	Policier
Pulp Fiction	1995	Policier
Play Time	1967	Comedie
Vertigo	1958	Suspense
Psychose	1960	Suspense
Parle avec elle	2002	Drame
Mon oncle	1958	Comedie
Volver	2006	Drame
La mauvaise éducation	2004	Drame

Une relation possède un nom, et se compose de colonnes désignées par un nom d'attribut avec des valeurs d'un certain domaine. Chaque ligne appelée tuple correspond à une entité. Le degré d'une relation est le nombre d'attributs (ou de colonnes) et le cardinal d'une relation est le nombre de tuples (ou de lignes).
Un schéma relationnel est constitué d'un ensemble de schémas de relations qui décrivent le contenu d'une relation.

Domaines

Un domaine de valeurs est un ensemble d'instances d'un type élémentaire (s'opposant au type structuré). Le système de types est figé et fourni par le système.

Attributs

Les attributs nomment les colonnes d'une relation. Ils servent à indiquer le contenu de cette colonne et à la référencer quand on effectue des opérations. Le nom d'un attribut peut apparaître dans plusieurs schémas de relations.

Schéma de relation

Un schéma de relation est simplement un nom suivi de la liste des attributs, chaque attribut n'étant présent qu'une seule fois et étant associé à son domaine. L'*arité* d'une relation est le nombre de ses attributs.

$$R(A_1 : D_1, A_2 : D_2, \ldots, A_n : D_n)$$

Instance d'une relation

Une instance d'une relation R se définit comme un sous-ensemble fini de produit cartésien des domaines des attributs de R (tous les tuples $(v_1,\ldots, v_n)$ où $v_i \in D_i$).

La définition d'une relation comme un ensemble entraîne que
- l'ordre des lignes n'a pas d'importance,
- on ne peut pas trouver deux fois la même ligne,
- et il n'y a pas de case vide.

Clé primaire d'une relation

C'est le plus petit sous-ensemble des attributs qui permet d'identifier chaque ligne de manière unique. La clé primaire est mise **en gras**. Le choix de la clé est très important pour la qualité du schéma.

Clé étrangère d'une relation

Attributs d'une relation qui correspondent à la clé primaire d'une autre (ou de la même) table.

Base de données

C'est un ensemble fini de relations. Le schéma de la base est l'ensemble des schémas des relations de cette base. La création d'un schéma de base de données est simple une fois qu'on

a déterminé toutes les relations qui constituent la base. Par contre, le choix de ces relations est primordial car il détermine en grande partie la qualité de la base. Le plus souvent, on conçoit le schéma à l'aide d'un modèle de données conceptuel (par ex, le modèle E/A), puis en transcrivant ce schéma en un schéma relationnel.

Contraintes d'intégrité

Ce sont des conditions à vérifier par toutes les relations : dépendances entre attributs, prédicat sur les valeurs d'un ou plusieurs attributs d'une relation, …

- contrainte de clé primaire : une relation possède une clé primaire
- contrainte de clé étrangère : une clé étrangère d'une relation a la même sémantique que la clé primaire correspondante
- contrainte d'intégrité référentielle : l'ensemble des valeurs d'une clé étrangère est inclus ou égal à l'ensemble des valeurs de la clé primaire correspondante
- contrainte existentielle : les valeurs d'un attribut sont définies pour tous les tuples de la relation
- contrainte de domaine : nécessité de précision sur le domaine des valeurs d'un attribut (liste ou intervalle de valeurs, formats, …)
- contrainte de valeur par défaut : nécessité d'une valeur à affecter à un attribut lors de l'insertion d'un tuple
- contrainte sur un tuple : prédicat (ou formule) mettant en jeu plusieurs attributs (ex : la date de naissance d'une personne est antérieure à la date de décès)
- contrainte de dépendance fonctionnelle : vérification des dépendances fonctionnelles entre groupes d'attributs
- contrainte de normalité : vérification du respect d'une forme normale
- contrainte de cardinalité : nécessité de borne pour le nombre de tuples d'une relation
- contrainte de type pré-post condition : condition à vérifier par le SGBD lors de l'évolution de la BD, sous forme d'une condition explicite avant, pendant ou après (ex : n'insérer un film que si le réalisateur est déjà dans la BD)
- règles actives : actions à exécuter suite à un changement d'état de la BD (dès qu'un cinéma est supprimé, il faut supprimer les salles correspondantes)

4. Passage d'un schéma E/A à un schéma relationnel

Dépendances fonctionnelles (DF)

Définition : Soient X et Y deux ensembles d'attributs. On dit que X détermine Y (ou Y dépend de X), noté $X \rightarrow Y$ lorsque le fait que deux tuples qui ont une même valeur sur les attributs de X implique qu'ils ont même valeur sur les attributs de Y ou encore qu'à toute valeur de X n'est associée qu'une et une seule valeur de Y, dans toute extension de la relation.

Propriétés des DF

- réflexivité : $X \rightarrow X'$ avec $X' \subseteq X$
- projection : $X \rightarrow Y,Z \Rightarrow X \rightarrow Y$ et $X \rightarrow Z$
- additivité : $X \rightarrow Y$ et $X \rightarrow Z \Rightarrow X \rightarrow Y,Z$
- augmentation : $X \rightarrow Y \Rightarrow X,Z \rightarrow Y$

- transitivité : $X \rightarrow Y$ et $Y \rightarrow Z \Rightarrow X \rightarrow Z$
- pseudo-transitivité : $X \rightarrow Y$ et $Y,W \rightarrow Z \Rightarrow X,W \rightarrow Z$

Théorème : toutes les DF peuvent être générées à partir d'un ensemble d'entre elles par les propriétés de réflexivité, augmentation et transitivité.
Typologie des DF $X \rightarrow Y$

- canonique : Y est réduit à un seul attribut
- triviale : $Y \subseteq X$
- élémentaire : il n'existe pas $X' \subset X$ tel que $X' \rightarrow Y$
- directe : il n'existe pas Z tel que $X \rightarrow Z$, $Z \rightarrow Y$ et $Z \rightarrow X$ n'existe pas

Hyper-graphe des DF
Sommets : attributs
Hyper-arcs : DF
La fermeture transitive d'un ensemble de DF consiste à appliquer systématiquement tant que cela est possible la propriété de transitivité.
La couverture minimale d'un hyper-graphe de DF est un ensemble de DF F vérifiant qu'aucune DF n'est redondante ($\forall$ DF f, F-{f}$\neq$F) et que toute DF est dans la fermeture transitive de F.

Formes normales

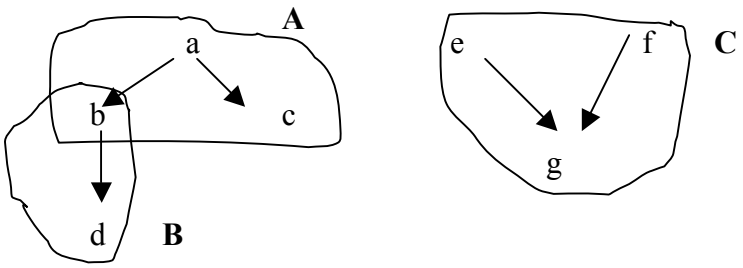
Notation : R (C_1, …, C_k, A_1, …, A_n) avec (C_1, …, C_k) clé primaire de R.

- Première forme normale (1FN) : les attributs d'une relation ne faisant pas partie de sa clé primaire sont fonctionnellement dépendants de la clé primaire de cette relation
  C_1, …, $C_k \rightarrow A_1$, …, A_n
- Deuxième forme normale (2FN = 1FN + élémentarité) : les attributs d'une relation ne faisant pas partie de sa clé primaire sont en DF élémentaire avec la clé primaire de cette relation
  $\neg\exists$ C $\subset$ {C_1, …, C_k} et A_j tel que $C \rightarrow A_j$
- Troisième forme normale (3FN = 2FN + non transitivité) : les attributs d'une relation ne faisant pas partie de sa clé primaire sont en DF élémentaire directe avec la clé primaire de cette relation (il n'y a pas de DF entre attributs hors clé)
  $\neg\exists$ A', A'' $\subset$ {A_1, …, A_n} et A' $\cap$ A'' = $\varnothing$ tel que $A' \rightarrow A''$
- Troisième forme normale de Boyce-Codd-Kent
  $\neg\exists$ A $\subset$ {A_1, …, A_n}, C $\subset$ {C_1, …, C_k} tel que $A \rightarrow C$
- 4FN, 5FN.

Théorème : toute relation admet une décomposition **sans perte ni redondance** en troisième forme normale.

Une fois l'ensemble des DF établi, le schéma relationnel se déduit de l'hyper-graphe des DF de couverture minimale. En effet, pour chaque sommet interne de l'hyper-graphe, on obtient une relation dont la clé primaire correspond au sommet interne et dont les attributs hors clé correspondent aux sommets fils du sommet interne ; de plus, les attributs correspondant à des sommets fils qui sont eux-mêmes des sommets internes sont des clés étrangères.

Exemple :

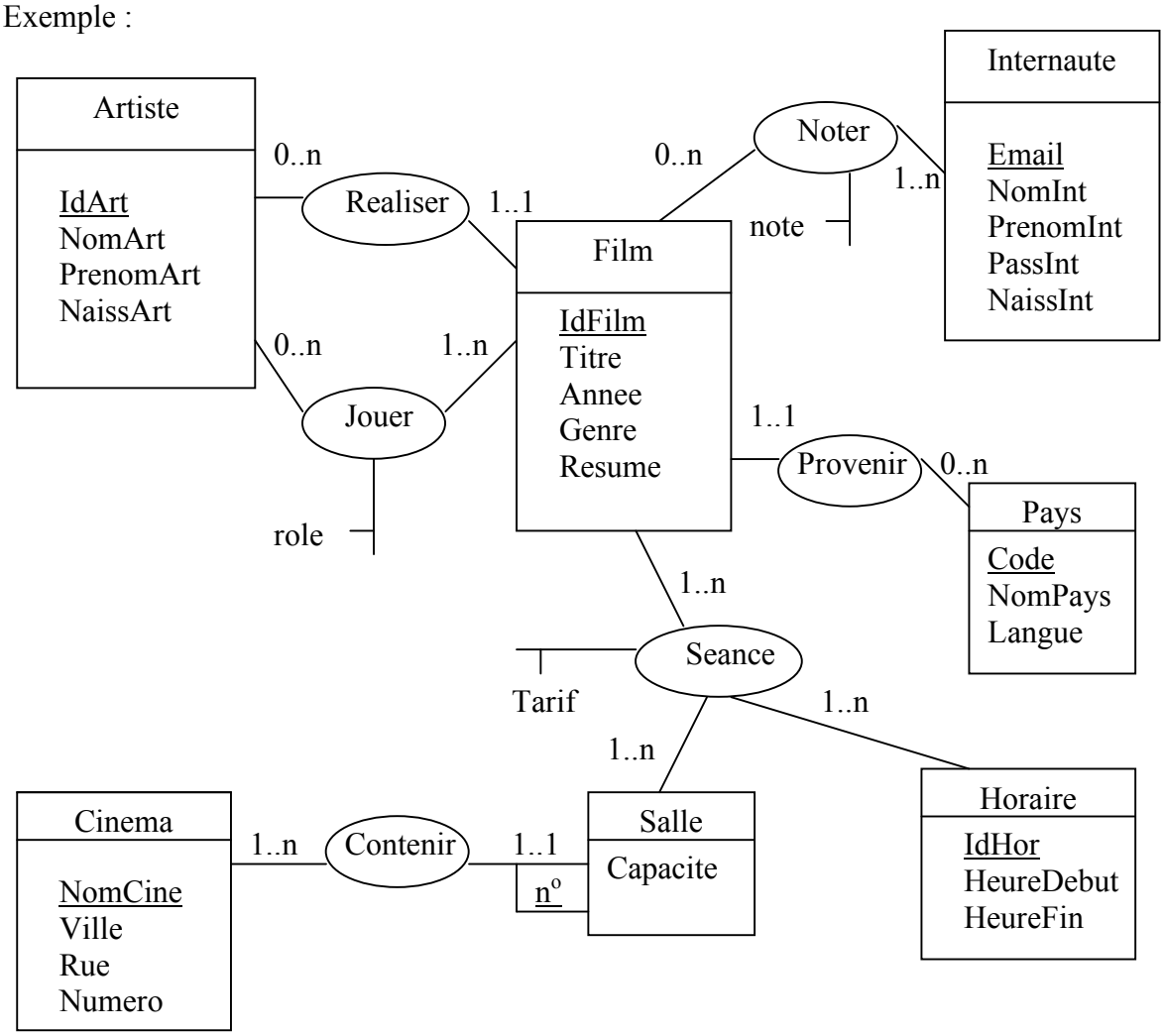


devient
A (a, b, c)

B (b, d)
C (e, f, g)
A est la clé de la relation A
B est la clé de relation B
(e, f) est la clé de la relation C
valeurs (A.b) ⊆ valeurs (B.b)

Règles de passage d'un schéma E/A à un schéma relationnel en 3FN

Exemple :



Règles

- Entité : pour chaque entité, on crée une relation de même nom que l'entité et chaque propriété de l'entité, y compris l'identifiant, devient un attribut de la relation et l'identifiant constitue la clé.
  Exemple :
  Film (**IdFilm**, Titre, Année, Genre, Résumé)
  Artiste (**IdArt**, NomArt, PrénomArt, NaissArt)
  Internaute (**Email**, NomInt, PrénomInt, PassInt, NaissInt)
  Pays (**Code**, NomPays, Langue)
  Cinéma (**NomCiné**, Ville, Rue, Numéro)
  Horaire (**IdHor**, HeureDébut, HeureFin)
- Entité faible : on crée un mécanisme de clé étrangère pour référencer l'entité forte dans l'entité faible. La clé étrangère est une partie de l'identifiant de l'entité faible.
  Exemple :
  Salle (**NomCiné, No**, Capacité)
- Association 1:n entre A et B : on crée les relations R_A et R_B correspondant aux entités A et B, l'identifiant de B devient un attribut de R_A. Une occurrence de A référence l'occurrence de B qui lui est associée à l'aide d'une clé étrangère.
  Exemple :
  Film (**IdFilm**, Titre, Année, Genre, Résumé, IdMES, Code)
  Le rôle tenu par l'artiste dans l'association disparaît. L'artiste dans Film a un rôle de réalisateur. Il n'est pas obligatoire que les clés étrangères aient le même nom que ceux de la clé primaire.
- Association binaire n:n[1] entre A et B : on crée les relations R_A et R_B correspondant aux entités A et B, on crée une relation R_{AB} pour l'association, la clé de R_A et la clé de R_B deviennent des attributs de R_{AB}, la clé de cette relation est la concaténation des clés de R_A et R_B et les propriétés de l'association deviennent des attributs de R_{AB}.
  Exemple :
  Rôle (**IdFilm, IdArt**, NomRôle)
  Notation (**IdFilm, Email**, note)
- Association ternaire : on atteint une des limites du modèle E/A en matière de spécification de contraintes. En première approche, on peut appliquer la règle énoncée pour les associations binaires généralisées.
  Exemple :
  Séance (**IdFilm, NomCiné, No, IdHor**, Tarif)
  On s'aperçoit que la même salle peut présenter deux films différents au même horaires. Si on souhaite éviter cette situation, la clé devient (**NomCiné, No, IdHor**), mais on ne respecte plus la règle de passage.
  Si on utilise la représentation d'une séance par une entité et des associations binaires, le problème reste le même : on peut avoir plusieurs films dans la même salle au même horaire.
  Séance (**IdSéance**, IdFilm, NomCiné, No, IdHor, Tarif)

  En cas d'association entre plus de 2 entités, la clé de la table représentant l'association est un sous-ensemble de la concaténation des clés. Il faut se poser soigneusement la question de la clé au moment de la création de la table car elle ne peut plus être déduite du schéma E/A.

[1] Cardinalités maximales

Choix des identifiants

Il est préférable en général de choisir un identifiant « neutre » qui ne soit pas une propriété de l'entité.
En effet :

1. Chaque valeur de l'identifiant doit caractériser de manière unique une occurrence.
2. Si on utilise un ensemble de propriétés comme identifiant, la référence à une occurrence est très lourde.
3. L'identifiant sert de référence externe et ne doit donc jamais être modifiable.

Le problème de l'identifiant « neutre » est qu'il ne donne pas d'indication sur l'occurrence qu'il réfère.

Dénormalisation du modèle logique

Deux objectifs principaux :

1. Simplifier le schéma relationnel en réduisant le nombre d'éléments qui le composent.
2. Faciliter l'accès aux données en introduisant un certain degré de redondance.

Ces techniques reviennent à introduire des anomalies dans le schéma. Il faut donc systématiquement comparer le gain attendu avec les risques courus.

- Suppression de relations : on peut supprimer les entités qui portent peu d'attributs en les déplaçant vers une autre relation.
  Exemple 1 : on supprime l'entité Cinéma.
  Salle (**NomCiné, No**, Ville, Rue, Numéro)
  L'adresse du cinéma est dupliquée autant de fois qu'il y a de salles.
  Exemple 2 : il est inutile de créer Horaire pour juste gérer un couple d'heures.
  Séance (**NomCiné, No, HeureDébut**, HeureFin, IdFilm, Tarif)
  En fait, la création d'un type d'entité Horaire ou Date présente plus d'inconvénients que d'avantages. En pratique, on place toujours un attribut horaire ou date dans le schéma de relation.
- Introduction de redondance : l'accès à une information peut être long ou compliqué et justifie l'introduction de redondances.
  Exemple : à partir de Séance, il faut consulter Salle puis Cinéma pour connaître l'adresse du cinéma ; il faut faire un calcul pour obtenir le nombre de salles d'un cinéma.
  Séance (**NomCiné, No, HeureDébut**, HeureFin, IdFilm, Ville, Rue, Numéro, Tarif)
  Cinéma (**NomCiné**, Ville, Rue, Numéro, NbSalles)

L'introduction de redondances présente le danger d'introduire des incohérences dans la base. On peut alors utiliser le mécanisme de triggers pour effectuer automatiquement la répercussion de la modification d'une donnée sur ses autres versions dans la base.