

Chapitre 9 : Bases de données

Table des matières

1	Concepts élémentaires	2
1.1	Introduction aux bases de données	2
1.1.1	Introduction	2
1.1.2	Systèmes de gestion de bases de données (SGBD) et paradigme logique . . .	2
1.1.3	Le modèle relationnel	2
1.1.4	Algèbre relationnelle	4
1.2	Le langage SQL : requêtes élémentaires	4
1.2.1	Gestion du contexte	4
1.2.2	Opération de projection	4
1.2.3	Opération de sélection	5
2	Conception de bases de données	6
2.1	Modèle entité-association	6
2.1.1	Introduction	6
2.1.2	Entités	6
2.1.3	Associations	6
2.1.4	Cardinalité d'une association	7
2.2	Passage du modèle entité-association au modèle relationnel	8
2.2.1	Idée	8
2.2.2	Cas des associations 1 – 1 et 1 – *	9

1 Concepts élémentaires

1.1 Introduction aux bases de données

1.1.1 Introduction

De nombreuses applications informatiques manipulent de grandes quantités de données qui doivent être organisées et stockées de sorte qu'il est possible de les traiter efficacement pour en ajouter, en retirer, ou en extraire de l'information. Ce traitement doit pouvoir se faire de manière concurrente tout en préservant l'intégrité des données.

1.1.2 Systèmes de gestion de bases de données (SGBD) et paradigme logique

Préserver l'intégrité des données est une tâche complexe dans un contexte d'accès concurrents, donc le traitement des données sera confié à un outil appelé SGBD dont le rôle est de recevoir les requêtes des utilisateurs (modification des données ou extraction d'information à partir de ces données) et de les traduire en des opérations effectuées sur la base de données. C'est au SGBD de garantir la cohérence des données au fur et à mesure des opérations réalisées. Les requêtes prennent en général la forme d'une description du résultat attendu sans indication sur la manière de calculer. C'est le SGBD qui implémente la recherche du résultat.

C'est le principe du **paradigme logique** : un programme est une description des propriétés que doit satisfaire le résultat. Un résultat est un jeu de paramètres qui satisfait les propriétés. Il n'y a aucune indication sur la manière de calculer les résultats.

Pour que cela fonctionne, la plupart des SGBD s'appuient sur un modèle introduit dans les années 1970, appelé modèle relationnel.

1.1.3 Le modèle relationnel

- Le modèle relationnel est un modèle mathématique basé sur la théorie des ensembles et la logique des prédicats, et qui présente les bases de données comme des objets qui définissent des relations entre les blocs d'information. C'est un modèle abstrait qui s'exprime indépendamment des implémentations possibles et qui couvre donc de nombreux SGBD.

- Dans ce modèle, une base de données est vue comme un ensemble de relations. Ces relations sont aussi appelées *tables* car on peut les représenter par des tableaux à double entrée dont les colonnes correspondent à un type d'information particulier. Ces colonnes sont appelées les *attributs* de la relation.

Exemple : dans le système d'information d'une bibliothèque, on peut avoir une table Document dont les attributs sont le titre, l'auteur, le genre, la date de parution, le nombre de pages, ...

- Chaque attribut est associé à un domaine qui correspond à l'ensemble des valeurs possibles par l'attribut. Le domaine permet de choisir un type pour implémenter concrè-

tement la base de données.

Étant donné une relation R dont les attributs sont A_1, \dots, A_n associés aux domaines D_1, \dots, D_n , on appelle *schéma relationnel* de R l'association des attributs et des domaines notée

$$R(A_1 : D_1, \dots, A_n : D_n)$$

Exemple : pour la table Document :

- Le titre et l'auteur sont des données textuelles ;
- Le genre est tirée d'une énumération finie (roman, poésie, théâtre, ...);
- La date de parution est une date ;
- Le nombre de pages est un entier.

D'où le schéma relationnel :

Document(titre : texte, auteur : texte, genre : enum(roman, ...), date de parution : date, nombre de pages : entier)

Le domaine "texte" peut être par exemple associé au type **string** des chaînes de caractères et le domaine entier au type **int**.

- Les lignes d'une relation de schéma $R(A_1 : D_1, \dots, A_n : D_n)$ correspondent aux éléments de la relation R , qui est un sous ensemble de $\prod_{k=1}^n D_k$. On appelle donc ces éléments des tuples ou des enregistrements.

Exemple pour la table Document :

Titre	Auteur	Genre	Date de parution	Nombre de pages
<i>La cousine Bette</i>	Honoré de Balzac	Roman	1846	240
<i>De la guerre</i>	Carl von Clausewitz	Traité	1832	240
<i>Cyrano de Bergerac</i>	Edmond Rostand	Théâtre	1857	280

Remarque : certains enregistrements peuvent coïncider pour certains attributs et on veut une manière efficace de les distinguer.

- On appelle *clé candidate* un ensemble minimal (pour l'inclusion) d'attributs permettant de caractériser de manière unique chaque enregistrement, *i.e* tel qu'il n'existe pas deux enregistrements qui coïncident sur tous les attributs de la clé.

Exemple : pour la table Document, {titre, auteur} devrait convenir.

Il peut y avoir plusieurs clés candidates et on doit en choisir une, appelée *clé primaire*. On souligne dans le schéma relationnel les attributs de la clé primaire pour les repérer efficacement.

Remarque : on choisit souvent d'ajouter un attribut entier pour numérotter les enregistrements, que l'on choisit comme clé primaire.

- Dans une relation R , on appelle *clé étrangère* un ensemble d'attributs qui constitue une clé candidate (souvent primaire) d'une autre relation.

Exemple : dans une table Emprunts décrivant les emprunts de la bibliothèque, on intègre la clé de la table Document pour identifier les documents empruntés.

1.1.4 Algèbre relationnelle

L'algèbre relationnelle est une théorie mathématique qui décrit des opérations que l'on peut réaliser sur une base de données du point de vue du modèle relationnel. Les propriétés qui découlent de cette théorie définissent un fondement rigoureux aux implémentations de SGBD en justifiant les optimisations des requêtes des utilisateurs. Cette théorie des H.P, mais nous l'étudierons *via* le langage de requêtes SQL (Structured Query Language).

1.2 Le langage SQL : requêtes élémentaires

1.2.1 Gestion du contexte

- Choix de la base de données : `USE <nom_base>;`
- Obtention du schéma relationnel d'une table : `DESCRIBE <nom_table>;`

Exemple :

```
1 || USE bibliotheque;  
2 || DESCRIBE Document;
```

Les opérations de création / suppression / modification de bases de données / de tables sont H.P.

1.2.2 Opération de projection

Pour visualiser le contenu d'une table, on utilise la requête `SELECT * FROM <nom_table>;`. C'est un cas particulier de l'opération de projection qui permet de construire une table ne contenant que les valeurs des tuples que pour certains attributs.

Attention, la table construite par une requête est éphémère.

Syntaxe : `SELECT <attribut1>, ..., <attributn> FROM <nom_table>;`

Exemple :

```
1 || SELECT titre, date_de_parution FROM Document;
```

Remarque : cette requête peut créer une table contenant des doublons qui sont conservés par défaut. On utilise le mot-clé `DISTINCT` pour éliminer les doublons.

Exemple :

```
1 || SELECT DISTINCT titre, date_de_parution FROM Document;
```

Les enregistrements du résultat d'une projection sont *a priori* rangés dans le même ordre que dans la table initiale. On peut choisir de les réordonner en utilisant le mot-clé `ORDER BY` suivi d'une liste d'attributs.

L'ordre associé à cette liste est l'ordre croissant lexicographique.

Exemple :



```
1 || SELECT * FROM Document
2 || ORDER BY auteur, date_de_parution;
```

Pour utiliser l'ordre décroissant selon l'un des attributs, on utilise le mot-clé DESC après le nom de l'attribut.

Exemple :

```
1 || SELECT * FROM Document
2 || ORDER BY date_de_parution DESC;
```

Attention, DESC ne porte que sur un seul attribut (le répéter si besoin).

Il est aussi possible de renommer les attributs du résultat d'une projection, ce qui peut être nécessaire dans des opérations plus complexes faisant intervenir des sous-requêtes ou plusieurs tables. On utilise pour cela le mot-clé AS.

Exemple :

```
1 || SELECT titre, auteur, date_de_parution AS date
2 || FROM Document;
```

On peut combiner tout cela :

```
1 || SELECT titre, date_de_parution AS date
2 || FROM Document
3 || ORDER BY date DESC, titre;
```

1.2.3 Opération de sélection

On peut ne conserver que les enregistrements qui satisfont une condition donnée *via* l'opération de sélection, introduite par le mot-clé WHERE.

Exemple :

```
1 || SELECT * FROM Document
2 || WHERE auteur = 'Edmond Rostand';
```

Les opérateurs de comparaison au programme sont : =, <>, <, >, <=, >=

Attention, les enregistrements ne fournissent pas forcément une valeur pour chaque attribut. Les attributs sans valeurs prennent la valeur spéciale NULL. On ne peut pas vérifier si un attribut a la valeur NULL avec les opérateurs ci-dessus : on utilise les opérateurs IS NULL et IS NOT NULL

Exemple :

```
1 || SELECT * FROM Document
2 || WHERE date_de_parution IS NULL;
```

On peut combiner plusieurs conditions grâce aux connecteurs logiques : AND, OR, NOT.

Exemple :

```
1 || SELECT * FROM Document
2 || WHERE NOT (date_de_parution >= 2000 AND nombre_de_page < 200);
```

Remarque : On peut combiner sélection et projection

On peut aussi fixer un nombre maximal d'enregistrements pour le résultat : le SGBD choisira les premiers qu'il traite jusqu'à éventuellement atteindre la borne. On utilise le mot clé LIMIT

Exemple : 10 ouvrages les + récents

```
1 | SELECT * FROM Document
2 | ORDER BY date_de_parution DESC
3 | LIMIT 10;
4 |
```

On peut ajouter un décalage lors de l'usage de LIMIT grâce au mot-clé OFFSET.

Exemple : dernier tiers du top 15 des auteurs ayant écrit les plus gros livres.

```
1 | SELECT DISTINCT auteur FROM Document
2 | ORDER BY nombre_de_pages DESC
3 | LIMIT 5 OFFSET 10;
4 |
```

2 Conception de bases de données

2.1 Modèle entité-association

2.1.1 Introduction

Le modèle *entité-association* est un modèle de la fin des années 1970 qui propose un formalisme pour décrire la structure des données en fonction des liens qu'elles entretiennent.

On lui associe une représentation graphique permettant d'identifier facilement les deux concepts de base du modèle : les *entités* et les *associations*.

2.1.2 Entités

Une entité est un objet donné pour lequel on dispose de données que l'on souhaite traiter.

Exemple : moi, le livre d'Edmond Rostand intitulé *Cyrano de Bergerac*, ...

Les informations associées à une entité sont ses attributs. On réunit un ensemble d'entités ayant des caractéristiques communes dans un type d'entité que l'on appelle aussi par abus entités.

Dans ce cadre, une entité concrète est appelée instance ou occurrence de l'entité.

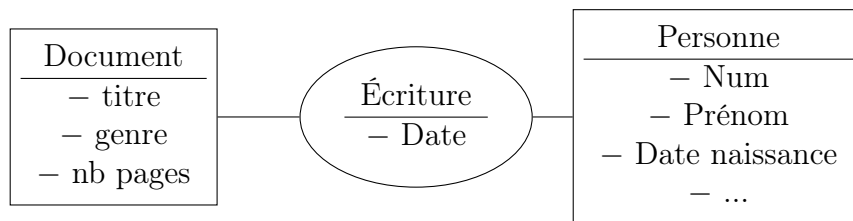
2.1.3 Associations

Une association exprime un lien entre plusieurs entités.

Exemple : Edmond Rostand *a écrit* le livre intitulé *Cyrano de Bergerac*.

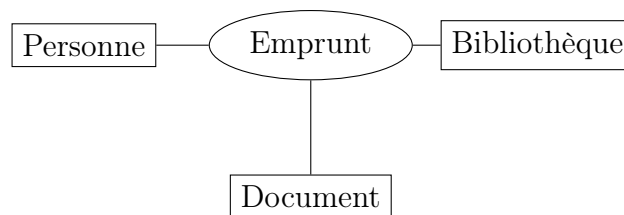
Une association peut aussi avoir des attributs (ex : date d'écriture) et on fait également l'abus de langage qui consiste à appeler association un type d'association.

Représentation graphique : on dessine deux types de bloc (les entités et les associations) et on lie les blocs d'entité *via* les blocs d'association qui expriment ces liens

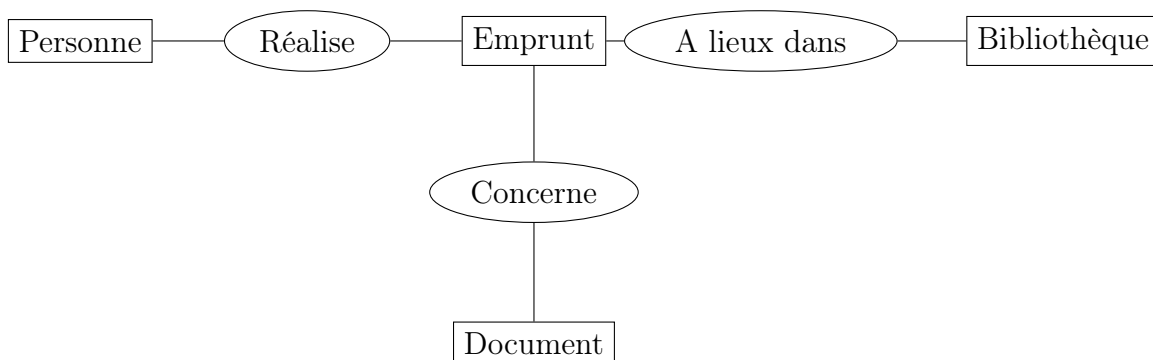


On distingue les associations binaires (*i.e* qui lient 2 entités) des associations n -aires. On peut toujours se limiter aux associations binaires en remplaçant une association n -aire par une nouvelle entité représentant l'association, liée aux n entités par n associations binaires.

Exemple :



donne :



2.1.4 Cardinalité d'une association

Le lien entre une entité et une association peut être étiqueté par un couple (p, q) représentant les nombres minimums et maximum de fois que l'entité peut apparaître dans une association de ce type : $q = *$ s'il n'y a pas de borne supérieure.

Exemple :



On appelle cette association une association $* - *$ car elle peut lier plusieurs entités à plusieurs autres.

D'autres types d'association sont :

– Les associations $1 - *$: elles lient une entité à plusieurs autres.

Exemple : une œuvre peut être tirée à plusieurs exemplaires mais un exemplaire n'est un tirage que d'une seule œuvre



– Les associations 1 – 1 : elles lient une entité à une seule autre.

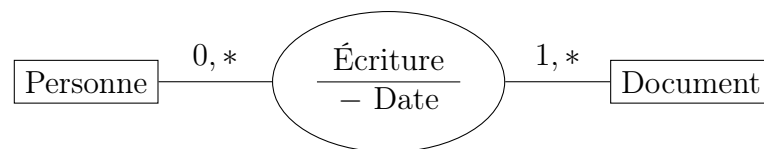
Exemple : chaque exemplaire d’une bibliothèque a une référence unique



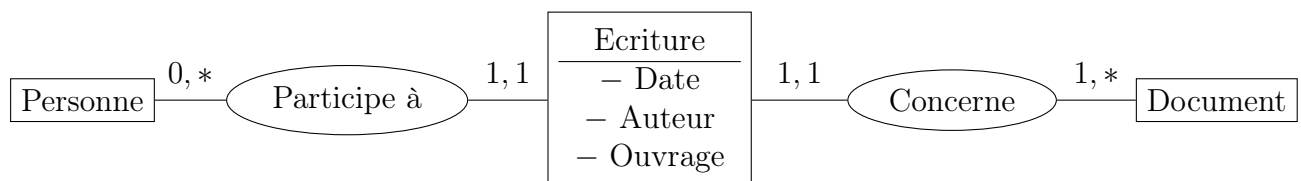
Remarque : on peut souvent fusionner les entités impliquées dans une association 1 – 1 sans introduire trop de redondance dans les données.

Une association * – * peut être scindée en deux associations 1 – * *via* l’introduction d’une nouvelle entité représentant l’association.

Exemple :



donne :



Remarque : les attributs de la nouvelle entité doivent permettre d’identifier les entités impliquées dans l’association : on utilise des clés étrangères dans le modèle relationnel associé à ce modèle.

2.2 Passage du modèle entité-association au modèle relationnel

2.2.1 Idée

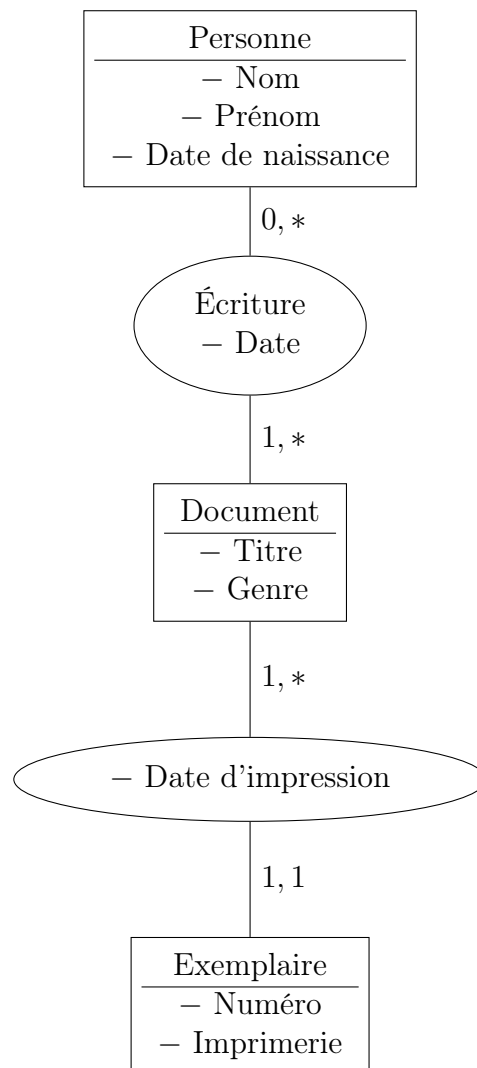
On associe une relation à chaque entité, avec les mêmes attributs, les associations expriment des liens entre les tables et introduisent donc des contraintes d’intégrité des données.

- La décomposition d’une association n -aire ou en associations binaires se traduit en la création d’une nouvelle entité donc d’une nouvelle association.
- La décomposition des associations binaires * – * en deux associations 1 – * se traduit aussi en la création d’une nouvelle table, dite de *jonction*.
- Après cela, il ne reste qu’à traiter les associations 1 – 1 et 1 – * .

2.2.2 Cas des associations 1 – 1 et 1 – *

Ces types d'association impliquent que pour l'une des deux entités, chaque instance de l'entité ne peut être associée qu'à une unique instance de l'autre entité. Pour exprimer ce lien entre les tables correspondantes on ajoute une clé étrangère à la première table, faisant référence à l'unique enregistrement de la seconde table impliquée dans l'association. Les attributs de l'association seront aussi intégrés à la première table.

Exemple :



Modèle relationnel associé : on utilise des identifiants numériques pour les clés primaires :

Personne(id : entier, nom : texte, prénom : texte, date de naissance : date)

Document(id Doc : entier, titre : texte, genre : enuml(...))

Exemple(numéro : entier, imprimerie : texte, id Doc : entier, date d'impression : date)

Écriture(date : date, id Doc : entier, id : entier)