Chapter 6

Le modèle relationnel

Toutes les applications web modernes exploitent des bases de données. Celles-ci assurent un stockage pérenne et des accès rapides. Elles permettent ainsi au serveur web de se concentrer sur la gestion des réponses aux requêtes envoyées par les navigateurs.

La structuration des bases de données est un point fondamental des application web qui sert l'expressivité mais aussi et surtout l'efficacité en temps et en espace. Une mauvaise structuration entraine d'importantes difficultés quant à l'expression de requêtes permettant d'obtenir de la connaissance sur les données. C'est aussi un goulet d'étranglement qui pose de réels problèmes de performance.

Ce chapitre introduit les concepts fondamentaux de la structuration des bases de données relationnelles en précisant les enjeux et la finalité.

6.1 Structuration des données

Les application web suivent souvent le même processus: récupérer des données brutes, les structurer afin qu'elles soient exploitables, et enfin proposer de la connaissance sur celles-ci. Les données brutes sont fournies par les utilisateurs par exemple lorsqu'on leur demande de saisir des formulaires. Dès qu'ils ont saisi leurs données brutes, l'application doit les structurer afin qu'elles soient exploitables. Une fois que les données sont structurées il est alors possible de les questionner pour obtenir de la connaissance.

Prenons l'exemple d'une application qui permet de gérer des photos. Celle-ci va demander à l'utilisateur qui veut ajouter une nouvelle photo de saisir le nom de la photo, son orientation (portrait ou paysage) et la date à laquelle elle a été prise. L'utilisateur saisie ces données mais elles n'ont pas encore beaucoup de sens car elles sont à l'état brut. L'application web va devoir transformer ces données et les structurer pour qu'elles soient exploitables. Elle va vérifier que l'orientation est soit 'portrait' soit 'paysage', que la date a été bien saisie, puis regrouper toutes les données pour former une description complète et cohérente d'une photo. Une fois structurées, ces données sont exploitables et permettent

d'obtenir de la connaissance. Par exemple, dès lors qu'on aura plusieurs photos de stockées dans la base de données, on pourra par exemple chercher toutes les photos orientées en mode portrait et qui ont été prises avant 1900.

La mise en place d'une bases de données nécessite de définir une structuration qualitative des données. Par qualitative on entend que les connaissances puissent être exprimées facilement, calculées rapidement et que les données nécessaires à leur établissement prennent le moins de place possible. S'il existe plusieurs façons de structurer les données (liste, tableau, arbre, graphe, etc.), les bases de données relationnelles ont pris le partie de n'utiliser que des tableaux avec des valeurs élémentaires : on appelle ces tableaux les tables de la base de données. Ce choix est motivé par la simplicité de la structure et la facilité de manipulation des données. En effet, dès lors qu'on a défini la structure des tables, on peut facilement et rapidement accéder aux données.

Définition 14 *Une base de données relationnelle structure les données sous forme de tables avec des valeurs élémentaires (booléen, nombre, chaîne de caractères de taille bornée, date, etc.).*

6.2 Les tables

La structuration d'une base de données relationnelle passe par la définition des tables contenues dans la base. Une table représente un concept cohérent de l'application, elle a un nom qui correspond à ce concept. Par exemple, dans le cas d'une application de gestion de photos, on peut définir la table des Photos, la table des Photographes, la table des Albums-Photos, etc.

Une table contient un ensemble fini de colonnes. Chaque colonne a un nom et est typée par un type de base (booléen, nombre, chaîne de caractères dont la taille est bornée, date, etc.). Il est important de noter qu'il n'est pas possible de mettre une valeur complexe dans une colonne (pas de liste, de tableau, d'arbre, etc.) Les colonnes des tables représentent les attributs du concept représenté par la table. Par exemple, dans la table des Photographes, on peut définir les colonnes nom et prénom, toutes les deux ayant un type chaine de caractères dont la taille ne dépasse pas 30 caractères (le choix de 30 est ici arbitraire, il devrait permettre la saisie de tous les noms et tous les prénoms possibles). Ces deux colonnes (nom, prénom) définissent ce qu'est un photographe.

Une ligne d'une table représente une entité unique. Par exemple, dans la table des Photographes, chaque ligne représente un photographe. Il n'est pas possible d'avoir deux lignes identiques car cela voudrait dire qu'on stocke deux fois la même entité.

Enfin, pour bien différencier chaque entité, une table doit définir une ou plusieurs colonnes qui serviront de clé pour distinguer les entités des unes des autres. La ou les colonnes qui servent de clé sont appelées clé primaire de la table. Si on ne trouve pas de clé dans les colonnes existantes, une façon simple consiste à ajouter une nouvelle colonne qui servira de clé (colonne souvent nommée id).

Photographes		
id	prénom	nom
1	Etienne	Carjat
2	Robert	Doisneau
3	Yann	Arthus-Bertrand

Table 6.1: La Table des Photographes

Définition 15 *Une table a un nom et possède un ensemble fini de colonnes qui ont chacune un nom et un type de base. Une ou plusieurs colonnes de la table servent de clé pour distinguer les entités des unes des autres. Chaque ligne de la table représente une entité unique.*

La table des Photographes de notre application exemple est illustrée par la Table 6.1. On peut voir que cette table est composée de 3 colonnes dont une est la clé primaire (la colonne id). On a en effet ajouté cette colonne car ni le prénom ni le nom ne peuvent servir de clé dans cette table (certaines personnes ont le même prénom, d'autres le même noms et il y a même des homonymes). Dans l'exemple, 3 entités sont enregistrées dans la table. Chacune de ses entités a un id différent.

6.3 Les associations entre les tables

Les concepts manipulées par une applications web sont souvent associées entre eux. Dans notre exemple d'application de gestion de photos, chaque photo est associée à un photographe et chaque photographe est associé à plusieurs photos.

La structuration d'une base de données doit permettre de représenter ces associations. Pour cela, on utilise des clés étrangères qui permettent de définir des liens entre deux tables.

Définition 16 *Une clé étrangère est une colonne d'une table qui référence une clé primaire d'une autre table.*

La multiplicité d'une association doit être prise en compte pour savoir où placer les clés étrangères. Il existe deux multiplictés importantes : 1 vers Plusieurs (noté 1-*), et Plusieurs vers Plusieurs (noté *-*).

Une association de multiplicité 1-* entre une table A et une table B (noté A 1-* B) est une association où une entité de la table A est associée à plusieurs entités de la table B, mais où une entité de la table B ne peut être associée qu'à une seule entité de la table A. Dans notre exemple d'application de gestion de photos, un photographe peut être associé à plusieurs photos mais une photo ne peut être associée qu'à un seul photographe. On a donc l'association Photographes 1-* Photos.

Dans le cas d'une association 1-*, la clé étrangère est placée dans la table qui contient les entités référencées plusieurs fois. Pour notre exemple, la clé

Photos			
id	nom	année	photographe
1	Charles Baudelaire	1863	1
2	Le baisé de l'hôtel de ville	1950	2
3	Les frères	1936	2
4	Coeur de Voh	1992	3

Table 6.2: La Table Photo

Albums-Photos	
id	nom
1	Incontournables
2	Noir et Blanc

Table 6.3: La Table des Albums Photo

étrangère est donc placée dans la table des Photos. Grâce à la clé étrangère, chaque entité de la table des Photos référence une entité de la table des Photographes. Il est donc possible que plusieurs entités de la table des Photos référencent la même entité de la table des Photographes (un photographe peut donc être associé à plusieurs photos). La Table 6.2 illustre cette association. La colonne photographe de la table des Photos est une clé étrangère qui référence la clé primaire id de la table des Photographes. Grâce à cette clé étrangère, on peut retrouver le photographe associé à une photo. Par exemple, on peut voir que Robert Doisneau est le photographe du baisé de l'hôtel de ville.

Une association de multiplicité *-* entre une table A et une table B (noté A *-* B) est une association où une entité de la table A est associée à plusieurs entités de la table B et où une entité de la table B est associée à plusieurs entités de la table A. Dans notre exemple d'application de gestion de photo, un album photo peut être associé à plusieurs photos et une photo peut être associée à plusieurs albums photo (on parle ici de photos virtuelles). On a donc l'association Albums-Photos *-* Photos.

Dans le cas d'une association *-*, il faut définir une nouvelle table qui représente l'association entre les deux tables associées, puis placer deux clés étrangères dans cette nouvelle table (une vers chacune des deux tables associées). Pour notre exemple, la table Albums-Photos est présentée dans la Table 6.3. Elle contient deux albums photo (Incontournables et Noir et Blanc). La Table 6.4 est la table qui représente l'association Albums-Photos *-* Photos. Cette table contient deux colonnes qui sont deux clés étrangères (une vers Albums-Photos et l'autre vers Photos). Les entités de cette table expriment les associations entre les albums photo et les photos. On voit que l'album photo Incontournables est associé aux photos 2 et 4. L'album photo Noir et Blanc est, quant à lui, associé aux photos 1, 3 et 4.

Albums-Photos-2-Photos	
album-photo	photo
1	2
1	4
2	1
2	2
2	3

Table 6.4: La Table Albums-Photos-2-Photos

Photos				
id	nom	année	orientation	photographe
1	Charles Baudelaire	1863	portrait	1
2	Le baisé de l'hôtel de ville	1950	paysage	2
3	Les frères	1936	portrait	2
4	Coeur de Voh	1992	paysage	3

Table 6.5: La Table des Photos avec la colonne orientation

6.4 Optimisation

La structuration d'une base de données ne s'arrête pas à la définition des tables et de leurs associations. Il faut également optimiser la base de données pour qu'elle soit la plus efficace possible. Un des points importants est de réduire autant que possible la redondance des données. On dit qu'une donnée est redondante si elle est répliquée plusieurs fois dans la base de données. Cela pose alors des problèmes d'efficacité (en place et en accès) mais aussi des problèmes d'intégrité car il faut s'assurer que les données redondantes sont cohérentes notamment lors de leurs modifications.

Reprenons l'exemple de la table des Photos mais en y ajoutant la colonne orientation dont le type est chaîne de caractères dont la taille ne dépasse pas 10 caractères (voir Table 6.5). On voit que les données relatives à l'orientation sont redondantes car on répète les informations sur chacune des entités. Cela peut poser des problèmes car la structure de la base de données ne peut garantir l'intégrité des données car elle n'interdit pas de mettre une valeur non valide dans la colonne orientation (n'import quelle chaîne de caractères).

On peut optimiser cette base de données en créant une nouvelle table (la table des Orientations) qui contient toutes les orientations possibles (voir Table 6.6). Grâce à cette table on peut mieux préciser les orientations des photos dans la Table Photo. Plus précisément, il s'agit d'une association Orientations 1-* Photos. On peut mettre en place cette association en considérant que la colonne orientation de la Table Photo est maintenant une clé étrangère vers la clé primaire de la table Orientation (voir Table 6.7). Cette optimisation permet

Orientations	
id	orientation
1	portrait
2	paysage

Table 6.6: La Table des Orientations

Photos				
id	nom	année	orientation	photographe
1	Charles Baudelaire	1863	1	1
2	Le baisé de l'hôtel de ville	1950	2	2
3	Les frères	1936	1	2
4	Coeur de Voh	1992	2	3

Table 6.7: La Table des Photos avec orientation qui est une clé étrangère vers la Table des Orientations

de gagner de la place mais aussi de garantir le fait que l'orientation ne peut être que portrait ou paysage tel que cela est défini dans la table Orientation.

6.5 Formalisation

Les principes de structuration des bases de données sont définis de manière formelle avec une base théorique issue des mathématiques. Nous présentons dans cette section des définitions plus rigoureuses des concepts que nous venons de présenter dans les sections précédentes.

D'un point de vue formel, une base de données structure les données sous forme de relations. Le terme relation est utilisé ici dans son sens mathématique : étant donné des domaines $D_1, D_2, \ldots, D_n$ (pas nécessairement distincts), R est une relation sur ces n domaines si elle est un ensemble fini de nuplets appartenant au produit cartésien des domaines ($D_1 \times D_2 \ldots \times D_n$). Une relation est donc sous ensemble fini du produit cartésien des domaines. Par exemple, les données contenues dans table Photo sont définies formellement par le produit cartésien : $(id \times nom \times date)$. Un élément de la relation est le nuplet $(1, 'CharlesBaudelaire', 1863)$.

Définition 17 *Etant donné des domaines $D_1, D_2, \ldots, D_n$ Une relation est un sous ensemble fini du produit cartésien $D_1 \times D_2 \times \cdots \times D_n$*

Les relations sont des concepts abstraits. Pour les présenter facilement on leur donne un nom et on nomme chacun de leurs domaines : c'est ce qu'on appelle un schéma relationnel.

Définition 18 *Un schéma permet de représenter une relation. Il précise le nom de la relation ainsi que les noms et les types de ses domaines (appelés attributs). On note* $R(A_1 < D_1 >, A_2 < D_2 >, \ldots, A_n < D_n >)$ *le schéma relationnel qui a pour nom* R *et qui définit plusieurs attibuts de nom* A_i *et de type* D_i*.*

Le concept de schéma relationnel permet de définir formellement le concept de clé primaire. En effet, une clé primaire est un ensemble d'attributs d'un schéma relationnel qui permet d'identifier de manière unique chaque nuple de la relation.

Définition 19 *Une clé primaire un attribut ou une combinaison d'attributs qui satisfait:*

- *une contrainte d'unicité : chaque valeur clé désigne de manière unique un nuplet de la relation ; autrement dit, deux nuplets ne peuvent pas posséder des valeurs identiques pour le(s) attribut(s) de la clé.*
- *une contrainte de minimalité : si une clé est composée d'un ensemble d'attributs, cette combinaison doit être minimale (aucun attribut ne peut en être retiré sans violer la règle d'unicité).*
- *Lorsqu'une relation possède deux ou plusieurs clés primaires non redondantes, l'une d'entre elles est choisie arbitrairement et appelée clé primaire de cette relation.*

Enfin, le concept de clé étrangère est lui aussi défini formellement grâce au concept de schéma relationnel.

Définition 20 *Lorsqu'un attribut ou groupe d'attributs d'un schéma relationnel* S *référencent la clé primaire d'un autre schéma* T*, on dit qu'on a une clé étrangère de* S *vers* T*. Une clé étrangère définit une contrainte d'association entre les relations des schéma* S *et* T*. Cette contraint maintient l'intégrité référentielle entre les deux relations.*

Les définitions que nous venons de présenter permettent de définir formellement les concepts fondamentaux des bases de données relationnelles. Nous allons définir les propriétés des dépendances fonctionnelles et les propriétés des dépendances fonctionnelles transitives qui permettent de mieux comprendre les objectifs d'optimisation des bases de données.

Définition 21 *Pour un schéma relationnel* $R(A_1 < D_1 >, A_2 < D_2 >, \ldots, A_n < D_n >)$*, un sous-ensemble de ses attributs* S*, et un attribut quelconque* A_i*.* A_i *dépend fonctionnellement de* S *si, pour tous les éléments dans la base, on connait la valeur de* A_i *en connaissant les valeurs de* S*. On note* $S \rightarrow A_i$ *la dépendance fonctionnelle de* A_i *par rapport à* S*.*

Définition 22 *Les dépendances fonctionnelles sont transitives. Si* $X \rightarrow Y$ *et si* $Y \rightarrow Z$ *alors* $X \rightarrow Z$

D'un point de vue formel, l'optimisation d'une base de données vise à faire en sorte qu'elle respecte des contraintes précisées dans des formes normales. Nous nous intéressons ici au trois première formes normales des bases de données. Celles-ci posent formellement les bases d'une structuration qualitative des données.

Définition 23 *Une relation est dite de première forme normale, si elle possède au moins une clé primaire et si toutes les valeurs sont élémentaires.*

Définition 24 *Une relation est dite de deuxième forme normale, si elle est de première forme normale, et que chaque attribut n'appartenant à aucune clé candidate est en dépendance totale de chaque clé candidate de la relation.*

Définition 25 *Une relation est dite de troisième forme normale si elle est en deuxième forme normale et si tout attribut n'appartenant pas à une clé n'a pas de dépendance fonctionnelle avec un autre attribut n'appartenant pas à une clé. C'est à dire encore que toutes les dépendances fonctionnelles vers des attributs n'appartenant pas à une clé, sont issues d'une clé candidate.*

En résumé, une base de donné qui respecte la troisième forme normale est bien formée dans le sens où toutes ses valeurs sont élémentaires, toutes les relations possèdent au moins une clé primaire, et tous les attributs sont en dépendance fonctionnelle d'une clé primaire.

6.6 SQL

La structure d'une base de données relationnelle peut être décrite en utilisant le langage SQL (Structured Query Language, en français langage de requête structurée). SQL est un langage standard mais chaque système de gestion de base de données (SGBD) peut avoir des variations qui lui sont propres. Nous allons nous intéresser ici à la syntaxe de Postgres qui est un SGBD libre et gratuit.

Avant de créer les tables, il faut commencer par construire une base de données. Il faut noter qu'un système de gestion de base de données permet de créer plusieurs bases de données. Chacune est identifiée par un nom. Le listing 6.1 présente la commande SQL permettant de créer la base de données nommée *application_photo*.

```
1 CREATE DATABASE application_photo;
```

Listing 6.1: creation d'une base de données en SQL

Une fois que la base est créée il est possible de créer les tables. La création d'une table se fait en donnant le nom de la table, les attributs et leur type. Les types SQL sont CHAR pour une chaîne de caractères de longueur fixe, VARCHAR pour une chaîne de caractères de longueur bornée, TEXT pour une chaîne de caractères, DEC pour un décimal, INT pour un entier, REAL pour un réel à virgule flottante dont la représentation est binaire, BOOLEAN pour vrai ou faux et DATE pour une date.

Le listing 6.2 présente la commande SQL permettant de créer la table *photographes*. On voit que les attributs *prenom* et *nom* sont de type $VARCHAR(100)$ et que l'attribut *prenom* ne peut pas être vide.

```
1 CREATE TABLE photographes (
2     prenom VARCHAR(100) NOT NULL,
3     nom VARCHAR(100)
4 );
```

Listing 6.2: creation d'une table en SQL

Postgres propose aussi le type SERIAL qui permet de créer un attribut entier qui est automatiquement incrémenté à chaque insertion d'une nouvelle ligne dans la table. Ce type est très pratique pour créer des clés primaires. Le listing 6.3 présente la commande SQL permettant de créer la table *photographes* avec une clé primaire.

```
1 CREATE TABLE photographes (
2     id SERIAL PRIMARY KEY,
3     prenom VARCHAR(100) NOT NULL,
4     nom VARCHAR(100)
5 );
```

Listing 6.3: creation d'une table en SQL avec une clé primaire

La création d'une table avec une clé étrangère doit se faire après la création de la table ciblée par la clé étrangère. Il faut préciser dans la commande SQL que l'attribut est une clé étrangère et qu'il référence une clé d'une autre table. Le listing 6.4 présente la commande SQL permettant de créer la table *photos* avec une clé primaire et une clé étrangère vers la table *photographes*.

```
1 CREATE TABLE photos (
2     id SERIAL PRIMARY KEY,
3     nom VARCHAR(100) NOT NULL,
4     annee DATE,
5     orientation VARCHAR(10),
6     photographe_id int references photographes(id)
7 );
```

Listing 6.4: creation d'une table en SQL avec une clé primaire et une clé étrangère

Dès lors que les tables sont construites, il est alors possible d'insérer des lignes dans la base de données. Le listing 6.5 présente les insertions de trois photographes et trois photos.

```
1 INSERT INTO photographes (prenom, nom) VALUES ('Etienne', 'Carjat')
      ;
2 INSERT INTO photographes (prenom, nom) VALUES ('Robert', 'Doisneau'
      );
3 INSERT INTO photographes (prenom, nom) VALUES ('Yann', 'Arthus-
      Bertrand');
4 INSERT INTO photo (nom, annee, orientation, photographe) VALUES ('
      Charles Baudelaire', '1863-01-01', 'portrait', 1);
5 INSERT INTO photo (nom, annee, orientation, photographe) VALUES ('
      Le baise de l hotel de ville', '1950-01-01', 'paysage', 2);
6 INSERT INTO photo (nom, annee, orientation, photographe) VALUES ('
      Les freres', '1936-01-01', 'portrait', 2);
7 INSERT INTO photo (nom, annee, orientation, photographe) VALUES ('
      Coeur de Voh', '1992-01-01', 'paysage', 3);
```

Listing 6.5: insertion de trois photographes et de trois photos avec les liens entre eux

6.7 Ce qu'il faut retenir

Ce chapitre présente les concepts fondamentaux des bases de données relationnelles Trois points sont à retenir:

- Les bases de données relationnelles stockent les données dans des tables. Une table définit un concept cohérent de l'application. Les colonnes d'une table définissent les attributs du concepts.
- Les tables sont reliées entre elles par des clés étrangères. C'est la multiplicité de l'association entre deux tables (1-* ou *-*) qui permet de définir où placer les clés étrangères.
- La formalisation des bases de données se base sur le concept de relation mathématiques. L'optimisation des bases de données vise à respecter des contraintes exprimées dans des formes normales. La troisième forme normale est la forme normale de base pour une structuration de qualité.