CAS PARTICULIER DES ASSOCIATIONS AVEC UNE CARDINALITÉ

L'approche relationnelle présentée par E. F. Codd possède un fondement mathématique rigoureux qui lui assure sa robustesse : le modèle relationnel est de loin le plus répandu dans le monde des bases de données.

Ce chapitre présente le concept de relation fondamental du modèle relationnel, ainsi que les opérations qui lui sont associées. Puis on aborde les méthodes qui permettent de passer du modèle conceptuel vu au chapitre précédent (entité-association ou UML) à un ensemble de relations. La qualité des relations ainsi produites est contrôlée par leur conformité aux trois premières formes normales. Ce processus peut conduire à une réorganisation des relations sans perte d'information. La manière de résoudre les incohérences à l'aide de la forme normale de Boyce-Codd est également présentée. Enfin, le lien entre les bases de données et la logique du premier ordre est rapidement exposé, afin de décrire la méthode d'interrogation de base de données par QBE (*Query By Example*) qui en découle.

1 Concepts

Cette section expose la notion de relation et la terminologie qui lui est associée. On présente ensuite les différents types de contraintes associées au contenu d'une relation, ainsi que la notion de clé d'une relation.

1.1 Modèle relationnel

Le modèle relationnel tire son nom de la notion de **relation** mathématique entre des éléments. Chacun de ces éléments peut prendre des valeurs dans un ensemble défini.

On suppose que l'on considère les appareils electroménagers d'une cuisine. Ils peuvent être contenus dans l'ensemble des valeurs suivantes : *réfrigérateur*, *cuisinière*, *hotte*, *robot*, *lave-vaisselle*. On considère par ailleurs un ensemble de couleurs qui peuvent être contenues dans l'ensemble des valeurs suivantes : *rouge*, *bleu*, *vert*, *jaune*, *blanc*, *noir*, *rose*, *jaune*. Les combinaisons possibles entre les appareils et les couleurs sont au nombre de 40, puisqu'il y a 5 appareils que l'on peut associer à 8 couleurs. Parmi toutes ces combinaisons possibles, on effectue une sélection qui représente par exemple la description d'une (horrible) cuisine dans le monde réel. Ces couples de valeurs choisis représentent les **faits** de la vie réelle.

```
(réfrigérateur, rouge)
(robot,mauve)
(cuisinière,jaune)
(lave-vaisselle,rouge)
```

Cet ensemble de couples de valeurs liées entre elles, que l'on nomme **tuples** dans le modèle relationnel, représente la **relation** entre les éléments 'appareil' et 'couleur'. Un tuple est aussi désigné par les termes « **nuplets** » ou « **enregistrements** ». On désigne également les éléments constitutifs de ces couples par les termes « **attributs** » ou « **champs** ».

On peut écrire formellement la relation de la manière suivante : **ma_cuisine(appareil, couleur)**. Cette écriture représente le **schéma relationnel** de la relation 'ma_cuisine'. Les valeurs énoncées précédemment pour les champs représentent leurs **domaines**, c'est-à-dire les ensembles de toutes les valeurs possibles pour un champ.

Une relation est totalement décrite par :

- le schéma relationnel ;
- les domaines des différents champs ;
- les tuples qui la constituent.

Le nombre de champs de la relation s'appelle son **degré de la relation**. Ici, la relation 'ma_cuisine' est de degré 2. Le nombre de tuples se nomme la **cardinalité** de la relation. La relation 'ma_cuisine' est de cardinalité 4. Attention, il ne s'agit pas de la même cardinalité que pour le modèle entité-association vu précédemment.

On représente une relation par une **table**, correspondant à la notion de tableau. Les tuples correspondent aux lignes et les colonnes aux champs de la relation. Voici sous forme de table une représentation de l'exemple précédent (voir figure 3.1).

Dans le modèle relationnel, la relation est l'élément fondamental. Toutes les opérations sur une ou plusieurs relations retourneront une relation. Un ensemble de relations reliées entre elles par des liens sémantiques constitue une **base de données**.

Figure 3.1

Table 'ma_cuisine'.

Appareil	Couleur
Réfrigérateur	Rouge
Robot	Mauve
Cuisinière	Jaune
Lave-vaisselle	Rouge

Notion de clé d'une relation et dépendance fonctionnelle

Lorsque que l'on utilise une base de données, il est nécessaire d'accéder à un enregistrement par le contenu d'un ou de plusieurs champs. On nomme **clé** d'une relation un champ, ou un ensemble de champs, d'une relation qui permet d'identifier précisément un enregistrement. Une relation peut comprendre plusieurs clés possibles ; ce sont les **clés candidates**. La clé choisie doit être minimale, c'est-à-dire qu'elle doit contenir le minimum de champs. Cette clé minimale ainsi définie est appelée la **clé primaire** de la relation. Une clé doit toujours contenir une valeur et celle-ci doit être unique pour chacun des enregistrements de la relation.

La clé ne peut être déduite simplement à partir du contenu de la relation ; on ne peut préjuger du contenu futur des enregistrements. Si l'on prend la relation 'ma_cuisine' vue précédemment, le champ 'Appareil' semble être une clé puisqu'il contient une valeur unique pour chacun des enregistrements. Cependant, il est tout à fait possible que la cuisine considérée comprenne un autre réfrigérateur de couleur bleue, auquel cas la valeur ne serait plus unique et ne permettrait pas de retrouver l'enregistrement. Dans ce cas de figure, la combinaison 'Appareil' 'Couleur' pourrait sembler être une clé, mais on ne peut en être certain compte tenu de l'évolution des données.

Pour désigner une clé primaire, il faut donc également prendre en compte le « sens » des données dans la vie réelle. Les relations qui existent entre les différents champs d'une relation vont être importantes : on exprime ces relations à l'aide de **dépendances fonctionnelles**. Une dépendance fonctionnelle existe entre deux ensembles de champs si les valeurs contenues dans l'un des ensembles de champs permettent de déterminer les valeurs contenues dans l'autre ensemble.

Cette propriété se traduit en termes mathématiques de la manière suivante :

```
Soit Cx un ensemble de champs (x1,x2,x3) et Cy un ensemble de champs (y1,y2,y3)
d'une relation R(d1,d2,x1,x2,x3,u1,u2,u3,y1,y2,y3).
Supposons que l'on considère des valeurs (x1_i, x2_i, x3_i) et (x1_j, x2_j, x3_j)
telles que l'on ait R(d1_i,d2_i,x1_i,x2_i,x3_i,u1_i,u2_i,u3_i,y1_i,y2_i,y3_i) et
R(d1_j,d2_j,x1_j,x2_j,x3_j,u1_j,u2_j,u3_j,y1_j,y2_j,y3_j).
On dit que Cy dépend fonctionnellement de Cx lorsque pour tout i et j si (x1_i, x2_i,
x3_i) est égal à (x1_j, x2_j, x3_j) alors (y1_i,y2_i,y3_i) est égal à
(y1_j,y2_j,y3_j).
```

Les dépendances fonctionnelles expriment la relation de hiérarchie qui existe entre les champs.

On considère l'exemple de table suivant (voir figure 3.2), qui correspond à la relation Lecteur(Numero_carte, Nom, Age, Ville, Etablissement). Cet exemple modélise les lecteurs d'une bibliothèque.

Figure 3.2

Relation 'Lecteur'.

Numero _carte	Nom	Age	Ville	Etablissement
1	Henri	10	Paris	Université Sorbonne
2	Stanislas	34	Paris	Université Jussieu
3	Henriette	44	Lyon	CHU Bron
4	Dominique	19	Nancy	Université Poincaré
5	Isabelle	56	Nancy	INPL
6	Olivier	51	Marseille	Université Saint Charles
7	Henri	98	Paris	Université Sorbonne
8	Jerome	23	Nancy	INPL
9	Laurence	34	Bordeaux	Université Victor Segalen
10	Christian	41	Paris	Ecole Normale Supérieure
11	Antoine	16	Marseille	Université Saint Charles
12	Laurence	34	Paris	Université Jussieu

Si l'on examine les données, on remarque qu'il ne peut y avoir de dépendances fonctionnelles entre les couples de champs (Ville, Etablissement) et les champs (Nom, Age). Il existe un enregistrement ('Laurence', '34') pour lequel les valeurs des champs (Nom, Age) correspondent à deux valeurs différentes de (Ville, Etablissement).

En revanche, on sait que, dans la réalité, un établissement est situé dans une ville et une seule (on le suppose pour cet exemple). Cela signifie qu'il existe une relation de dépendance entre les champs 'Etablissement' et 'Ville'. Le contenu des champs 'Ville' et 'Etablissement' des enregistrements de notre relation se conforment à cette relation de dépendance. A une valeur donnée de 'Etablissement' correspond bien une valeur unique de 'Ville'.

La valeur du champ 'Numero_carte' est unique pour chacune des personnes. On constate que ses valeurs sont identifiantes pour tous les autres champs de la relation. Chaque champ dépend fonctionnellement du champ 'Numero_carte'. Ses valeurs sont uniques et jamais vides : c'est une clé candidate. Dans cet exemple, c'est la seule clé possible car les autres champs n'ont jamais de valeur unique. Le champ' 'Numero_carte' est choisi comme clé primaire de la relation.

Remarque

Tous les champs qui ne font pas partie d'une clé candidate d'une relation possèdent des dépendances fonctionnelles avec cette clé.

Les dépendances entre les champs représentent les liens entre les différents éléments du monde réel. On rappelle qu'elles ne peuvent être déduites du contenu de la relation, même si cela peut constituer un point de départ. Elles sont donc déterminées durant la phase précédente d'analyse du monde réel. Il est possible en revanche de vérifier si les enregistrements présents dans la relation se conforment à ces dépendances. La détermination des dépendances fonctionnelles entre les champs est fondamentale pour l'étape de normalisation, traitée à la section « Normalisation ». Elles sont également à la base de la méthode dite de la « relation universelle », qui sera abordée dans la section « Normalisation ».

1.3 Cohérence des données et contraintes d'intégrité

Afin de garantir la cohérence des données pour l'ensemble des relations constitutives de la base de données, on applique des restrictions sur le contenu des données que l'on nomme **contraintes d'intégrité**. La vérification de la cohérence se situe à plusieurs niveaux :

- adapter le contenu des champs par rapport au sens des données dans le monde réel ;
- préserver la cohérence du contenu de l'ensemble des relations qui sont liées ;
- éliminer les problèmes d'incohérence dus à la redondance : en effet la duplication des données rend délicates la maintenance et l'évolution de la relation.

Adapter le contenu des données

La cohérence par rapport au sens des données est liée à la notion de domaine de valeurs du champ déjà abordée : on parle de **contrainte de domaine** ou **cohérence sémantique**. Par exemple, l'âge d'une personne ne peut être négatif ou excéder 120. On a précédemment défini une contrainte d'intégrité sur l'ensemble des champs qui constituent une clé primaire : une clé ne peut contenir de valeur nulles et ses valeurs doivent être uniques. On procède à la définition des contraintes lors de l'étape d'analyse du monde réel. Leur mise en œuvre effective interviendra au moment de la création de la relation et sera réalisée par le SGBD.

Préserver la cohérence des données

On rappelle qu'une base de données est constituée par un ensemble de relations reliées entre elles. Les contenus des champs capables de lier ces relations doivent être cohérents entre eux pour pouvoir effectuer l'opération de jointure. Si l'on considère l'exemple de la base de données exemple 'casse', on ne doit pas permettre la saisie d'une valeur identifiant une voiture dans la relation 'vente' qui n'existe pas dans la relation 'voiture'. On fait dans ce cas référence au contenu d'une colonne d'une autre relation, le champ 'NumVoit' de la relation voiture, pour contrôler le contenu du champ 'NumVoit' de la relation 'vente'. Le champ 'NumVoit' est alors une **clé étrangère** qui permet de réaliser la notion d'**intégrité référentielle**. De même que précédemment, on détermine ces références lors de la phase d'analyse du monde réel et le SGBD permet de les réaliser. On préfère appliquer en général ces contraintes non pas lors de la création des relations, mais plutôt lorsque les données ont déjà été insérées, en particulier dans les relations de références. Cette précaution évite les problèmes de références impossibles à résoudre entre les relations, ce qui provoque parfois l'incapacité à insérer des données.

Éliminer la redondance des données

Les incohérences provoquées par la **redondance** d'information représentent le principal souci du concepteur d'une base de données. En effet, lorsque les données sont dupliquées, aucun mécanisme ne peut garantir que le changement de la valeur d'une donnée est répercuté correctement sur les autres données. Dans l'exemple ci-dessus (voir figure 3.2) d'une relation des lecteurs de bibliothèque, on remarque la redondance d'information qui existe entre les champs 'Ville' et 'Etablissement'. Si le nom de l'établissement 'INPL' change, il faut mettre à jour toutes les lignes qui contiennent son nom. La redondance est mise en évidence par la dépendance fonctionnelle qui existe entre ces deux champs. Dans cet exemple, détecter la redondance est évident. Cependant, il s'agit généralement d'incohérences délicates à déceler lorsque le nombre de relations est élevé ou encore si le sujet modélisé par la base de données n'est pas familier. Les incohérences de ce type seront résolues par la réorganisation des relations lors de la phase de normalisation.

2 Opérations du modèle relationnel

La manipulation des données dans le modèle relationnel se fait à l'aide d'opérations formelles reposant sur des concepts mathématiques issus de la théorie des ensembles : c'est **l'algèbre relationnelle**. Les opérations de l'algèbre relationnelle portent sur une ou plusieurs relations (ou tables). Le résultat retourné par ces opérations est toujours une relation (ou table).

Cette section présente une liste non exhaustive des différentes opérations de l'algèbre relationnelle. Ces dernières peuvent être regroupées en plusieurs catégories :

- les opérations classiques issues de la théorie des ensembles (union, intersection, différence) ;
- les opérations plus spécifiques du monde relationnel (projection, sélection, jointure) qui constituent les opérations fondamentales de l'algèbre relationnelle ;
- les opérations de types calcul et agrégats qui ne constituent pas réellement des opérations fondamentales (ni ensemblistes, ni relationnelles) mais qui sont le complément indispensable des précédentes.

2.1 Opérations ensemblistes

L'algèbre relationnelle emprunte un certain nombre de concepts à la théorie des ensembles. À l'exception du produit cartésien, ces opérations ont la particularité de ne pouvoir s'utiliser que pour des relations qui ont exactement la même structure ou des structures compatibles. Elles sont toutes de type binaire, car elles s'appliquent à deux relations.

Union

L'opération d'union consiste en la mise en commun des enregistrements de chaque relation. Les enregistrements identiques ne sont intégrés qu'une seule fois. Dans l'exemple ci-après (voir figures 3.3 et 3.4), le tuple dont la valeur du champ 'Numero_carte' vaut 3 ne sera pas dupliqué. L'union est représentée par le caractère « ∪ ». Cette opération sert typiquement à la « consolidation » de données de même type provenant de différentes sources.

Figure 3.3

Relations 'Lecteur_1' et 'Lecteur_2'.

Lecteur_1(Numero_carte, Nom, Age, Ville, Etablissement)

Numero_carte	Nom	Age	Ville	Etablissement
1	Henri	10	Paris	Université Sorbonne
2	Stanislas	34	Paris	Université Jussieu
3	Henriette	44	Lyon	CHU Bron

Lecteur_2(Numero_carte, Nom, Age, Ville, Etablissement)

Numero_carte	Nom	Age	Ville	Etablissement
3	Henriette	44	Lyon	CHU Bron
4	Dominique	19	Nancy	Université Poincaré
5	Isabelle	56	Nancy	INPL

Figure 3.4 **Lecteur_1 ∪ Lecteur_2**

Union des relations 'Lecteur_1' et 'Lecteur_2'.

Numero_carte	Nom	Age	Ville	Etablissement
1	Henri	10	Paris	Université Sorbonne
2	Stanislas	34	Paris	Université Jussieu
3	Henriette	44	Lyon	CHU Bron
4	Dominique	19	Nancy	Université Poincaré
5	Isabelle	56	Nancy	INPL

Différence

L'opération différence consiste à désigner les enregistrements qui appartiennent à une relation sans appartenir à l'autre. La différence est représentée par le caractère « – » (voir figure 3.5). Attention, cette opération n'est pas symétrique ; le résultat de l'opération 'Lecteur_1 – Lecteur_2' est en général différent de 'Lecteur_2 – Lecteur_1'. Elle permet par exemple d'éliminer des enregistrements d'une relation par rapport à une liste.

Figure 3.5 **Lecteur_1 – Lecteur_2**

Différence des relations 'Lecteur_1' et 'Lecteur_2'.

Numero_carte	Nom	Age	Ville	Etablissement
1	Henri	10	Paris	Université Sorbonne
2	Stanislas	34	Paris	Université Jussieu

Intersection

L'opération intersection peut se déduire de la précédente ; elle désigne les enregistrements qui sont communs aux deux relations. L'intersection est représentée par le caractère « ∩ » (voir figure 3.6). Elle permet de trouver les éléments communs à deux relations.

Figure 3.6 **Lecteur_1 ∩ Lecteur_2**

Intersection des relations 'Lecteur_1' et 'Lecteur_2'.

Numero_carte	Nom	Age	Ville	Etablissement
3	Henriette	44	Lyon	CHU Bron

Produit cartésien

Le produit cartésien permet la combinaison des enregistrements de deux relations sans tenir aucun compte du contenu des données. Les relations n'ont donc pas besoin d'avoir la même structure. Le caractère représentant le produit cartésien est « X ».

Figure 3.7 **Relations 'ma_cuisine' et 'musicien'.**

ma_cuisine(Appareil, Couleur)

Appareil	Couleur
Réfrigérateur	rouge
Robot	mauve
Cuisinière	jaune

musicien(Nom, Instrument)

Nom	Instrument
Jaco Pastorius	Basse électrique
Bill Evans	Piano

L'exemple a été choisi de manière à montrer que les relations ne nécessitent pas de rapports entre elles pour faire un produit cartésien. Combiner des appareils de cuisine et des musiciens n'a aucun sens dans la réalité (voir figures 3.7 et 3.8).

Figure 3.8 **Produit cartésien des relations 'ma_cuisine' et 'musicien'.**

Le_resultat(Appareil, Couleur, Nom, Instrument) = ma_cuisine(Appareil, Couleur) X musicien(Nom, Instrument)

Appareil	Couleur	Nom	Instrument
réfrigérateur	rouge	Jaco Pastorius	Basse électrique
réfrigérateur	rouge	Bill Evans	Piano
robot	mauve	Jaco Pastorius	Basse électrique
robot	mauve	Bill Evans	Piano
cuisinière	jaune	Jaco Pastorius	Basse électrique
cuisinière	jaune	Bill Evans	Piano

2.2 Opérations relationnelles

Projection

La projection consiste à extraire certains champs de la relation, ce qui donne à cette dernière un degré inférieur à la relation de départ. Voici la relation obtenue à partir de la relation 'Lecteur' en projetant les champs 'Nom' et 'Ville' (voir figure 3.9).

Figure 3.9 **Projection des champs 'Nom' et 'Ville' de la relation 'Lecteur'.**

Lecteur_proj(Nom, Ville)

Nom	Ville
Henri	Paris
Stanislas	Paris

Nom	Ville
Henriette	Lyon
Dominique	Nancy
Isabelle	Nancy
Olivier	Marseille
Henri	Paris
Jerome	Nancy
Laurence	Bordeaux
Christian	Paris
Antoine	Marseille
Laurence	Paris

Intuitivement, dans une représentation de type table, on conserve uniquement les colonnes sur lesquelles la projection est faite.

Sélection ou restriction

La sélection consiste à extraire les enregistrements de la relation. On utilise des critères pour caractériser les enregistrements sélectionnés. Voici la relation obtenue à partir de la relation 'Lecteur 'en sélectionnant les enregistrements dont le contenu du champ 'Ville' est 'Marseille' (voir figure 3.10). La structure de la relation résultat est la même que celle de la relation de départ.

Figure 3.10 Sélection sur la relation 'Lecteur'.

Lecteur_sel(Numero_carte, Nom, Age, Ville, Etablissement)

Numero_carte	Nom	Age	Ville	Etablissement
6	Olivier	51	Marseille	Université Saint Charles
11	Antoine	16	Marseille	Université Saint Charles

Intuitivement, dans une représentation de type table, on conserve uniquement les lignes répondant au critère.

Jointure

La jointure est l'opération fondamentale de l'algèbre relationnelle qui permettra d'exprimer le sens du lien entre les relations dans le monde réel. La liaison entre les relations s'effectue par le contenu commun d'un champ. L'opération de jointure peut être vue comme une sélection des enregistrements obtenus par le produit cartésien des relations, dont les contenus du champ sur lequel on effectue la jointure sont égaux. On l'appelle dans ce cas une **équijointure**. Les champs dont on compare les contenus sont nommés **champs de jointure**.

On considère les deux relations Lecteur_bis(Numéro_carte, Nom, Num_Etablissement) et Etablissement(Num_Etablissement, Ville, Nom_Etablissement) dont le contenu suit (voir figure 3.11).

Figure 3.11 **Lecteur_bis(Numero_carte, Nom, Num_Etablissement)**

Relations 'Lecteur_bis' et 'Etablissement'.

Numero_carte	Nom	Num_Etablissement
1	Henri	1
2	Stanislas	2
3	Henriette	1

Etablissement(Num_Etablissement, Ville, Nom_Etablissement)

Num_Etablissement	Ville	Nom_Etablissement
1	Paris	Université Jussieu
2	Lyon	CHU Bron
3	Nancy	Université Poincaré
4	Paris	Université Sorbonne

Même si l'on ne dispose pas du modèle conceptuel associé, on constate que l'on peut relier ces deux relations par le champ 'Num_Etablissement'. Les informations concernant l'établissement de la relation Lecteur_bis sont stockées dans la relation Etablissement dont la clé est le champ Num_etablissement. Pour obtenir la liste des lecteurs ainsi que les informations concernant leur établissement, on effectue une jointure entre ces relations sur le champ 'Num_Etablissement' (voir figure 3.12).

Figure 3.12 **Lecteur_joint_Etablissement(Numero_carte, Nom, Num_Etablissement_1, Num_Etablissement_2, Ville, Nom_Etablissement)**

Jointure des relations 'Lecteur_bis' et 'Etablissement' sur le champ 'Num _Etablissement'.

Numero _carte	Nom	Num_Etablis-sement_1	Num_Etablis-sement_2	Ville	Nom _Etablissement
1	Henri	1	1	Paris	Université Jussieu
2	Stanislas	2	2	Lyon	CHU Bron
3	Henriette	1	1	Paris	Université Jussieu

Le champ 'Num_Etablissement' y figure deux fois car une occurrence vient de la relation 'Lecteur' et l'autre de la relation 'Etablissement'. Afin de ne conserver qu'une valeur du champ 'Num_Etablissement', on utilise l'opération de **jointure naturelle** (voir figure 3.13).

Figure 3.13 **Lecteur_jointnat_Etablissement(Numero_carte, Nom, Num_Etablissement, Ville, Nom_Etablissement)**

Jointure naturelle des relations 'Lecteur_bis' et 'Etablissement' sur le champ 'Num _Etablissement'.

Numero_carte	Nom	Num _Etablissement	Ville	Nom _Etablissement
1	Henri	1	Paris	Université Jussieu
2	Stanislas	2	Lyon	CHU Bron
3	Henriette	1	Paris	Université Jussieu

On peut considérer l'opération de jointure comme une sélection sur le produit cartésien des deux relations. On ne conserve que les lignes dont le contenu du champ sur lequel s'effectue la jointure est égal. Les lignes grisées sont celles qui sont sélectionnées lors d'une jointure (voir figure 3.14).

Figure 3.14

Produit cartésien des relations 'Lecteur_bis' et 'Etablissement'.

Produit_cartesien_Etablissement(Numero_carte, Nom, Num_Etablissement_1, Num_Etablissement_2, Ville, Nom_Etablissement)

Numero _carte	Nom	Num_Etablis-sement_1	Num _Etablissement	Ville	Num_Etablis-sement_2
1	Henri	1	1	Paris	Université Jussieu
1	Henri	1	2	Lyon	CHU Bron
1	Henri	1	3	Nancy	Université Poincaré
1	Henri	1	4	Paris	Université Sorbonne
2	Stanislas	2	1	Paris	Université Jussieu
2	Stanislas	2	2	Lyon	CHU Bron
2	Stanislas	2	3	Nancy	Université Poincaré
2	Stanislas	2	4	Paris	Université Sorbonne
3	Henriette	1	1	Paris	Université Jussieu
3	Henriette	1	2	Lyon	CHU Bron
3	Henriette	1	3	Nancy	Université Poincaré
3	Henriette	1	4	Paris	Université Sorbonne

On verra au chapitre consacré à SQL que c'est l'une des manières d'écrire une jointure.

Jointure externe

La jointure externe n'est pas réellement une opération de base de l'algèbre relationnelle. Elle est cependant nécessaire pour pouvoir répondre plus facilement à des questions du type « Quels sont les établissements qui n'ont pas de lecteurs ? » Il s'agit d'une opération de jointure étendue qui inclut dans le résultat les lignes n'ayant pas de correspondance sur le contenu du champ de jointure. Voici le résultat de la jointure externe de la relation Etablissement avec la relation Lecteur sur le champ 'Num_Etablissement' (voir figure 3.15). On met en correspondance les valeurs du champ 'Num_Etablissement' de toutes les lignes de la relation 'Etablissement' avec celles de la relation 'Lecteur'.

Figure 3.15

Jointure externe des relations 'Lecteur_bis' et 'Établissement' sur le champ 'Num _Etablissement'.

Etablissement_jointext_ Lecteur(Num_Etablissement_1, Ville, Nom_Etablissement, Numero_carte, Nom, Num_Etablissement_2)

Num_Etablissement_1	Ville	Nom _Etablissement	Numero _carte	Nom	Num_Etablissement_2
1	Paris	Université Jussieu	1	Henri	1
2	Lyon	CHU Bron	2	Stanislas	2
3	Nancy	Université Poincaré	NULL	NULL	NULL
4	Paris	Université Sorbonne	NULL	NULL	NULL

Ici, les valeurs 3 et 4 du champ 'Num_Etablissement_1' provenant de la relation Etablissement n'ont pas de correspondance dans la relation 'Lecteur'. Les lignes sont tout de même incluses dans le résultat mais les champs provenant de la relation 'Lecteur' prennent la valeur 'NULL'. Cette valeur signifie que le champ ne contient pas de valeur.

Pour répondre à la question « Quels sont les établissements qui n'ont pas de lecteurs ? », il nous suffit de sélectionner les lignes qui contiennent la valeur 'NULL', par exemple dans le champ 'Numéro_carte'. Dans un second temps, on effectue une projection sur le champ 'Nom_Etablissement' (voir figure 3.16).

Figure 3.16

Sélection et projection sur la jointure externe des relations 'Lecteur_bis' et 'Etablissement' sur le champ 'Num _Etablissement'.

Etablissement_jointext_Lecteur_selproj(Nom_Etablissement)

Nom _Etablissement
Université Poincaré
Université Sorbonne

La jointure externe n'est pas une opération **symétrique**. L'opération inverse, c'est-à-dire la jointure externe de la relation 'Lecteur' avec la relation 'Etablissement' sur le champ 'Num_Etablissement' donne dans ce cas le même résultat qu'une jointure naturelle. En effet, toutes les valeurs du champ 'Num_Etablissement' de la relation 'Lecteur' ont une correspondance dans la relation 'Etablissement'. C'est ce que l'on souhaite puisque la relation 'Etablissement' fait office de relation de référence pour le champ 'Num_Etablissement' de la relation 'Lecteur'. L'opération de jointure externe peut être utilisée pour détecter ce type d'incohérence.

2.3 Calculs et agrégats

Une règle dans le domaine des bases de données est que tout ce qui peut se calculer ne doit pas être stocké. On évite ainsi la perte de place et l'incohérence qui peut en découler suite au stockage d'informations redondantes. Les opérations de l'algèbre relationnelle présentées dans les sections précédentes ne permettent pas de créer de nouveau champ par calcul à partir du contenu d'autres champs sans utiliser un langage de programmation. Des

fonctions de calculs ont été définies afin de répondre à ce besoin. Elles seront détaillées au chapitre sur « SQL ».

On considère la relation La_boutique(Num_facture, Article, Prix, Quantite) [voir figure 3.17].

Figure 3.17 **La_boutique(Num_Facture, Article, Prix, Quantite)**

Relation 'La_boutique'.

Num_Facture	Article	Prix	Quantite
101	Bananes	12	45
1034	Choux	5	129
2345	Riz	4	60
0987	Gazelle	15	48

On suppose que l'on veut exemple trouver le total pour chaque facture en multipliant le prix par la quantité. Il est alors possible d'ajouter un champ 'Total' dont le contenu sera calculé par l'expression 'Prix' * 'Quantité' (voir figure 3.18).

Figure 3.18 **La_boutique_total(Num_FactureArticle, Prix, Quantite, Total)**

Relation 'La_boutique' avec le champ calculé 'Total'.

Num_Facture	Article	Prix	Quantite	Total
101	Bananes	12	45	540
1034	Choux	5	129	645
2345	Riz	4	60	240
0987	Gazelle	15	48	720

Il est également possible d'effectuer des opérations statistiques globales d'un champ, comme les calculs du nombre d'enregistrements, de moyenne, de maximum, de minimum. On obtient dans ce cas une nouvelle relation réduite à une ligne et une colonne qui contient la valeur calculée. En effet, le résultat d'une opération sur une relation est toujours une relation (voir figure 3.19).

Figure 3.19 **La_boutique_moyenne(Moyenne_Prix)**

Moyenne des prix de la relation 'La_boutique'.

Moyenne_Prix
9

De même, les opérations de base de l'algèbre relationnelle ne permettent pas de répondre à des questions du type : « Combien trouve-t-on de personnes par ville ? » La solution consiste alors à regrouper les enregistrements qui contiennent les mêmes valeurs dans le champ 'Ville' : ce regroupement s'appelle un **agrégat**. On applique ensuite à chaque sous-relation ainsi constituée une opération statistique, dans notre cas l'opération de comptage (voir figure 3.20).

Figure 3.20 **Lecteur_parville(Ville, Nombre)**

Agrégat par ville de la relation 'La_boutique'.

Ville	Nombre
Bordeaux	1
Lyon	1

Ville	Nombre
Marseille	2
Nancy	3
Paris	5

3 Passage du modèle conceptuel au relationnel

On a exposé au chapitre sur l'analyse du monde réel deux formalismes différents – entité-association et UML – pour représenter le modèle conceptuel obtenu. Il existe des règles simples pour passer de ces modèles à un ensemble de relations qui sera utilisable directement dans le SGBD. Cet ensemble de relations s'appelle le **schéma relationnel** et constitue le **modèle logique des données**. On présente dans cette section les règles de transition qui permettent d'exprimer le schéma relationnel à partir du modèle conceptuel des données. Pour illustrer ces règles, on utilise la terminologie du modèle entité-association, mais leur expression est aisément adaptable au formalisme UML.

3.1 Règles générales

Deux grandes règles générales s'appliquent toujours, quel que soit le cas. Les exceptions que l'on aborde ensuite permettent simplement d'obtenir un schéma plus compact :

- Une entité devient une relation composée des champs de l'entité. La clé de cette relation est la même que celle de l'entité.
- Une association devient une relation composée des deux clés des entités associées. L'ensemble de ces deux clés constitue celle de la relation. Lorsqu'elle en possède, les champs de l'association forment les champs « non clés » de cette nouvelle relation.

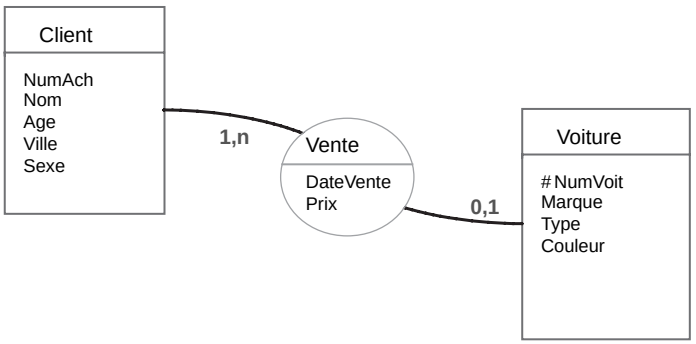


Figure 3.21
Modèle entité-association de la base de données 'casse'.

À partir du modèle entité-association de la base de données 'casse' élaboré au chapitre précédent (voir figure 3.21), on obtient trois relations en appliquant ces règles. Les entités 'voiture' et 'personne' deviennent les relations voiture(NumVoit, Marque, Type, Couleur) et personne(NumAch, Nom, Age, Ville, Sexe). L'association se transforme en relation vente(DateVent, Prix, NumAch, NumVoit).

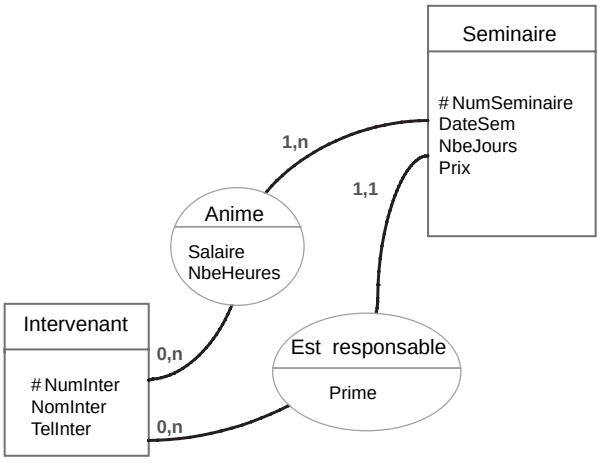
3.2 Cas particulier des associations avec une cardinalité de type '1-1'

Considérons le cas du modèle entité-association qui représente l'activité d'organisation de séminaires décrite très succinctement de la manière suivante :

- Un séminaire est animé par un ou plusieurs intervenants.
- Un séminaire ne possède qu'un seul responsable.

Figure 3.22

Modèle entité-association de la base de données 'animation de séminaire'.



Deux associations existent entre les deux entités 'Séminaires' et 'Intervenant' : 'Anime' et 'Est_responsable'. Les attributs utilisés pour les entités et les associations sont visibles sur le modèle entité-association (voir figure 3.22).

Si l'on applique les règles générales énoncées ci-dessus, on obtient quatre relations :

- Intervenant(NumInter, NomInter, TelInter)
- Seminaire(NumSem, DateSem, Prix, NbeJour)
- Anime(NumInter, NumSem, NbeHeure, SalaireHor)
- Est_responsable(NumInter, NumSem, Prime)

L'une des cardinalités de l'association 'Est_responsable' est de type '1-1', car un séminaire possède un responsable et un seul. De cette dernière propriété, on peut en déduire que la clé de la relation 'Est_responsable' n'est pas minimale. En effet, pour la relation 'Est_responsable' l'identifiant du séminaire détermine celui du responsable. En d'autres termes, on identifie une dépendance fonctionnelle entre les champs 'NumSem' et 'NumInter' pour la relation 'Est_responsable'. On choisit le champ 'NumSem' comme clé de la relation 'Est_responsable'.

La relation 'Est_responsable' devient ainsi Est_responsable(NumSem, NumInter, Prime).

Les relations 'Est_responsable' et 'Seminaire' ont alors la même clé, et il est possible de les regrouper. On obtient la relation Seminaire_Res(NumSem, DateSem, Prix, NbeJour, NumInter, Prime). Il s'est produit une sorte de « glissement » des éléments de l'association 'Est_responsable' vers l'entité 'Seminaire'.

On peut en déduire une règle complémentaire des règles générales précédentes :

- Lorsque l'association entre deux entités comprend une cardinalité de type '1-1', on ne crée pas de relation pour l'association. Les champs de l'association seront intégrés à la

relation créée pour l'entité qui est associée avec la cardinalité '1-1'. La clé de cette relation est toujours celle de l'entité.

Remarque

L'un des intérêts d'UML est de disposer de logiciels capables, à partir d'un modèle conceptuel exprimé en UML, de déduire automatiquement les relations en utilisant les règles énoncées ici. D'autres règles apparaîtraient dans le cas de l'utilisation des extensions objet que propose UML ; cependant ces possibilités dépassent le cadre de cet ouvrage.

4 Normalisation

Le modèle relationnel procure des outils destinés à tester la qualité et la cohérence des relations dans un schéma relationnel créé à l'étape précédente. Cette étape, appelée **normalisation**, permettra de vérifier certaines propriétés des relations et le cas échéant de les transformer.

On aborde dans cette section les trois premières formes normales qui suffisent dans la plupart des cas et qui permettent une décomposition du schéma relationnel sans perte d'information. La forme normale de Boyce-Codd qui détecte d'autres incohérences, mais propose une décomposition avec perte d'informations de dépendance, est présentée à la fin de la section. Il existe d'autres formes normales – la quatrième et la cinquième –, qui ne sont pas présentées dans cet ouvrage : ces dernières sont parfois difficiles à appréhender et les trois premières formes normales suffisent en général pour obtenir un schéma relationnel de qualité.

La normalisation d'un schéma relationnel suggère une autre méthode pour obtenir un ensemble de relations. On part d'une relation unique qui contient tous les champs, que l'on appelle la **relation universelle**. À l'aide des décompositions proposées par la mise en forme normale et du graphe des dépendances fonctionnelles des champs de cette relation, on parvient par raffinements successifs à un ensemble de relations normalisées. Cette méthode de la relation universelle est toutefois assez difficile à manipuler dès que l'on dépasse une taille critique du nombre de champs.

4.1 Première forme normale

La première forme normale s'intéresse au contenu des champs. Elle interdit la présence, appelée **multivaluation,** de plusieurs valeurs dans un même champ d'une relation. En effet, la notion de dépendance fonctionnelle entre les champs ne peut plus être vérifiée s'ils possèdent plusieurs valeurs. Elle s'exprime de la manière suivante :

Tout champ contient une valeur atomique.

Comment passer en première forme normale ?

La relation suivante n'est pas en première forme normale ; le champ 'Auteurs' contient plusieurs valeurs (voir figure 3.23).

Figure 3.23 **Publication(NumPubli, Titre, Auteurs)**

Relation 'Publication' avec un champ multivalué.

NumPubli	Titre	Auteurs
13490	Le vin et l'avenir	Jean Lasso, Hubert De la Tuque, Stanislas Wilski
21322	Bière et progrès social	Aristide Salem, Jean Lasso, Salome Dupont
45333	Le champagne et la France	Penelope Light, Vanessa Martinez, Salome Dupont

Une solution pour résoudre ce problème est de décomposer le champ 'Auteurs' en 'Auteur_1', 'Auteur_2', 'Auteur_3', 'Auteur_4', etc. Ainsi, la relation est bien en première forme normale. L'ennui est que l'on ne sait pas à l'avance le nombre d'auteurs que peut posséder une publication, et le problème consiste donc à savoir combien de champs ajouter.

La solution plus correcte, mais également plus lourde à mettre en œuvre, est de décomposer cette relation en trois relations : Publication(NumPubli, Titre), Auteur(NumAuteur, Nom, Prenom) et EstEcrite(NumPubli, NumAuteur). Ces trois relations sont la représentation de la réalité « une publication est écrite par des auteurs ». Elle se modélise par deux entités 'Publication' et 'Auteur' reliées par l'association 'EstEcrite', comme on l'a vu au chapitre 2, « Analyse du monde réel ».

On obtient alors le résultat suivant (voir figure 3.24).

Figure 3.24 **Publication(NumPubli, Titre)**

Décomposition de la relation 'Publication' en trois relations.

NumPubli	Titre
13490	*Le vin et l'avenir*
21322	Bière et progrès social
45333	Le champagne et la France

Auteur(NumAuteur, Nom, Prenom)

NumAuteur	Nom	Prenom
1	Lasso	Jean
2	De la Tuque	Hubert
3	Wilski	Stanislas
4	Salem	Aristide
5	Dupont	Salome
6	Light	Penelope
7	Martinez	Vanessa

EstEcrite(NumPubli, NumAuteur)

NumPubli	NumAuteur
13490	1
13490	2
13490	3
21322	4
21322	1
21322	5
45333	6
45333	7
45333	5

On doit alors effectuer des jointures sur les différentes relations afin de reconstituer l'information dans son intégralité. Cette décomposition en trois relations se fait sans perte d'information.

4.2 Deuxième forme normale

Bien évidemment, une relation doit déjà se trouver en première forme normale pour être en deuxième forme normale. Cette dernière recherche la redondance d'information dans une relation. Elle interdit les dépendances fonctionnelles possibles entre les champs qui composent la clé et les autres champs. On peut l'exprimer de la manière suivante :

La relation est en première forme normale.

Tout champ qui n'appartient pas à la clé ne dépend pas d'une partie de la clé.

Comment passer en deuxième forme normale ?

La solution consiste à décomposer la relation en deux relations. La nouvelle relation créée a pour clé la partie de la clé dont dépendent les autres champs qui constituent ses champs.

On considère la relation 'Produit' suivante (voir figure 3.25).

Figure 3.25 Relation 'Produit'.

Produit(Article, Fournisseur, Adresse, Prix)

Article	Fournisseur	Adresse	Prix
Marteau	SOGENO	Paris	5
Tournevis	ARTIFACT	Lille	10
Tournevis	SOGENO	Paris	23
Pince	LEMEL	Paris	34
Mètre	ARTIFACT	Lille	24

La clé est constituée des champs 'Article' et 'Fournisseur'. Or, il y a une relation de dépendance entre le champ 'Fournisseur', qui est une partie de la clé, et le champ 'Adresse'.

On décompose alors la relation pour éliminer la redondance ainsi créée. La nouvelle relation

aura pour clé la partie de la clé de la relation d'origine dont dépendent fonctionnellement les autres champs. Dans cet exemple, il s'agit du champ 'Fournisseur'. Les autres champs dépendants constituent le reste de la relation. Il s'agit ici du champ 'Adresse'.

On obtient alors le résultat suivant (voir figure 3.26).

Figure 3.26 **Décomposition de la relation 'Produit' pour passer en deuxième forme normale.**

Produit(Article, Fournisseur, Prix)

Article	Fournisseur	Prix
Marteau	SOGENO	5
Tournevis	ARTIFACT	10
Tournevis	SOGENO	23
Pince	LEMEL	34
Mètre	ARTIFACT	24

Fournisseur(<u>Fournisseur</u>, Adresse)

Fournisseur	Adresse
SOGENO	Paris
ARTIFACT	Lille
LEMEL	Paris

Comme précédemment, il est nécessaire de faire une jointure pour reconstituer l'information. La décomposition en deux relations se fait sans perte d'information.

Remarque

Si la clé d'une relation est atomique, c'est-à-dire composée d'un seul champ, elle est naturellement en deuxième forme normale.

4.3 Troisième forme normale

La troisième forme normale recherche également la redondance d'information dans une relation. On cherche s'il existe une dépendance entre deux champs qui ne font pas partie d'une clé. Si c'est le cas, on se trouve dans la situation où un champ dépend d'un autre champ qui dépend lui même d'une clé. La clé considérée peut être primaire ou secondaire. La troisième forme normale interdit donc les dépendances fonctionnelles dites « **transitives** » entre les champs. Elle s'exprime de la manière suivante :

La relation est en deuxième forme normale (donc en première forme normale).

Tout champ n'appartenant pas à une clé ne dépend pas d'un autre champ non clé.

Comment passer en troisième forme normale ?

La solution est également de décomposer la relation de départ en deux relations. La nouvelle relation créée a pour clé le champ dont dépendent les autres champs qui constituent ainsi la dépendance transitive.

On considère la relation suivante (voir figure 3.27).

Figure 3.27 **Baladeur(NumBal, Marque, Type, Couleur)**

Relation 'Baladeur'.

NumBal	Marque	Type	Couleur
12	Apple	Ipod	Blanc
43	Creative	Zen	Noir
23	Apple	Ipod	Noir
29	Creative	Zen	Gris
34	Sony	MZ-RH910	Rouge

La clé de cette relation est un numéro, 'NumBal', car il peut y avoir dans notre stock plusieurs baladeurs de même marque, de même type et de même couleur. Les marques déposent les noms des objets qu'elles fabriquent de façon à les identifier sur le marché. Il existe donc une dépendance fonctionnelle entre le champ 'Type' (qui n'appartient pas à la clé) et le champ 'Marque' (qui n'appartient pas à la clé). On décompose la relation en en créant une nouvelle qui a pour clé le champ dont dépendent les autres champs constituant la dépendance transitive. Il s'agit dans ce cas du champ 'Type'. Les autres champs de la nouvelle relation sont composés des champs qui en dépendent fonctionnellement : ici, le champ 'Marque' (voir figure 3.28).

Figure 3.28 **Baladeur(NumBal, Type, Couleur)**

Décomposition de la relation 'Baladeur' pour passer en troisième forme normale.

NumBal	Type	Couleur
12	Ipod	Blanc
43	Zen	Noir
23	Ipod	Noir
29	Zen	Gris
34	MZ-RH910	Rouge

Baladeur_type(Type, Marque)

Type	Marque
Ipod	Apple
MZ-RH910	Sony
Zen	Creative

Comme précédemment, il est nécessaire de faire une jointure pour reconstituer l'information dans son intégralité. La décomposition en deux relations se fait sans perte d'information.

Remarque

Les deuxième et troisième formes normales traitent des problèmes différents. L'ordre dans lequel on les considère pour la normalisation, mise en deuxième forme puis en troisième forme normale, est plutôt lié à l'historique qu'à une nécessité réelle.

4.4 Forme normale de Boyce-Codd

La forme normale de Boyce-Codd traite un cas un peu différent de ceux de la deuxième et troisième forme normale. Il s'agit du cas où une partie d'une clé dépend d'un champ. Comme pour la troisième forme normale, la clé considérée peut être une clé primaire ou secondaire. Une relation en troisième forme normale n'est pas toujours en forme « Boyce-Codd », mais l'inverse est toujours vrai.

Tout champ appartenant à une clé ne dépend pas d'un autre champ non clé.

Comment passer en forme normale de Boyce-Codd ?

Lors de la décomposition de la relation en deux relations, plusieurs choix sont envisageables compte tenu des liens de dépendances multiples entre les différents champs. On choisit la décomposition qui permet de reconstituer strictement l'information d'origine sans générer de données supplémentaires. La perte d'une dépendance fonctionnelle est fréquente lors de cette opération. La relation créée a pour clé la partie de celle-ci dont dépendent les autres champs constituant la dépendance.

On considère la relation suivante (voir figure 3.29).

Figure 3.29 **Dictaphone(Marque, Produit, Prix, Couleur)**

Relation 'Dictaphone'.

Marque	Produit	Prix	Couleur
Philips	LD 1024	49	Blanc
Olympus	VN 1664	49	Noir
Philips	LD 5647 H	59	Blanc
ImaginR	VN 1664	69	Gris
Olympus	VN 234 PC	79	Rouge

La clé de cette relation est constituée par les champs 'Marque' et 'Produit'. En effet, un produit est fabriqué sous licence par la société ImaginR et a donc le même nom que celui proposé par la société Olympus. Il est alors nécessaire d'utiliser les deux champs pour constituer la clé. Pour se démarquer les unes des autres, les sociétés utilisent des couleurs personnalisées destinées à identifier la marque : Philips, le blanc et l'orange ; Olympus, le rouge et le noir ; ImaginR, le gris. On a donc une relation de dépendance entre ces deux champs. La relation est en troisième forme normale, mais elle n'est pas en forme de Boyce-Codd.

Plusieurs décompositions sont possibles : par exemple Dictaphone(Marque, Type, Prix) et Marque_coul(Couleur, Marque). Mais cette décomposition génère des tuples non désirés au moment de la jointure. Quant à la décomposition Dictaphone(Type, Prix, Couleur) et Marque_coul(Couleur, Marque), elle permet de reconstituer l'information de départ par une jointure sur le champ 'Couleur' (voir figure 3.30).

Figure 3.30 **Dictaphone(Produit, Prix, Couleur)**

Relation 'Dictaphone' décomposée pour passer en forme de Boyce-Codd.

Produit	Prix	Couleur
LD 1024	49	Blanc
VN 1664	49	Noir
LD 5647 H	59	Blanc

Produit	Prix	Couleur
VN 1664	69	Gris
VN 234 PC	79	Rouge

Marque_coul(Couleur, Marque)

Couleur	Marque
Blanc	Philips
Noir	Olympus
Gris	ImaginR
Rouge	Olympus

On a perdu la dépendance de 'Couleur' par rapport à 'Marque', 'Produit'.

5 Logique du premier ordre et base de données

Dans cette section, le fondement logique de l'approche relationnelle est brièvement abordé pour présenter une méthode d'interrogation : les QBE (*Query By Example*). La logique du premier ordre a pour but de formaliser le raisonnement naturel en considérant la déduction comme le résultat d'un calcul. Elle a fait l'objet de recherches intenses depuis l'Antiquité : Aristote, Frege et Gödel pour n'en citer que quelques-uns sont parmi les plus prestigieux à s'être penchés sur la question. Les principes de la logique du premier ordre, dans une version restreinte que l'on appelle **le calcul des propositions**, ont trouvé de nombreuses applications en informatique. Après quelques rappels sur le formalisme de la logique, on aborde ses liens avec l'approche relationnelle en base de données et l'on présente la méthode d'interrogation par QBE.

5.1 Rappels sur la logique du premier ordre

Le formalisme de la logique du premier ordre permet d'exprimer des phrases dans un langage non ambigu. Il offre la possibilité, par des règles de dérivation, ou réécritures, de déduire d'autres phrases. L'élément fondamental de la logique du premier ordre est le **prédicat** qui exprime le lien entre différents éléments. La validité de ce prédicat est généralement restreinte à un ensemble de valeurs que l'on nomme **domaine du discours**.

```
mange(x,y) sera par exemple vrai pour les valeurs du couple (x,y) suivantes
(homme,navet)
(lapin,carotte)
(homme,lapin).
```

On utilise classiquement des variables (x,y dans notre exemple) et des constantes dans les expressions. Les prédicats peuvent être associés par des connecteurs logiques : $\wedge$ (et), $\vee$ (ou), $\neg$ (non), $\Rightarrow$ (implique). Le domaine de discours des variables s'exprime à partir des quantificateurs $\forall$ (quel que soit) et $\exists$ (il existe). Ce formalisme simple permet d'exprimer des propriétés comme celle-ci : « si x est mangé par y ou y est mangé par x, ils appartiennent tous deux à la même chaîne alimentaire ».

```
∀ x ∃ y mange(x,y) ∨ mange(y,x) ⇒ meme_chaine_alimentaire(x,y)
```

Cette formule peut se réécrire en utilisant des règles de déduction dans le but d'aboutir à un ensemble de clauses plus aisées à vérifier : c'est sur ce principe que fonctionne la démonstration automatique de théorèmes. La différence entre un (bon) mathématicien et un programme est évidemment que le mathématicien va choisir d'emblée la série de règles adaptée pour parvenir plus rapidement au résultat.

Voici une possibilité de réécriture de la formule précédente.

a ⇒ b donne ¬a ∨ b
∀ x ∃ y (¬(mange(x,y) ∨ mange(y,x)) ∨ (meme_chaine_alimentaire(x,y)))

¬ (a ∨ b) donne ¬a ∧ ¬b
∀ x ∃ y (¬mange(x,y) ∧ ¬ ? mange(y,x)) ∨ (meme_chaine_alimentaire(x,y)))

(¬a ∧ ¬b) ∨ c donne (¬a ∨ c) ∧ (¬b ∨ c)
∀ x ∃ y ((¬mange(x,y) ∨ meme_chaine_alimentaire(x,y)) ∨ (¬(mange(y,x) ∨ meme_chaine_alimentaire(x,y)))

¬a ∨ b donne a ⇒ b
∀ x ∃ y ((mange(x,y) ⇒ meme_chaine_alimentaire(x,y)) ∨ (mange(y,x) ⇒ meme_chaine_alimentaire(x,y))

∀ x ∃ y signifie que y est défini en fonction de x, on remplace y par F(x)

On obtient ainsi deux clauses reliées par ∨, on l'appelle clause de Horn qui représente la fin de la réécriture :

```
mange(x,F(x)) ⇒ meme_chaine_alimentaire(x,F(x)))
mange(F(x),x) ⇒ meme_chaine_alimentaire(x, F(x))
```

Dans notre cas, on peut s'assurer assez facilement que ces deux implications sont toujours vérifiées pour le domaine du discours.

5.2 Lien avec les bases de données

Le lien avec ce qui a été énoncé précédemment sur le modèle de bases de données relationnelles est le suivant :

- Le prédicat correspond à la relation.
- Le domaine du discours correspond au contenu de la relation.

Ce formalisme permet d'exprimer les questions, naturellement ambiguës du langage parlé, par une formule. Cette formule est réécrite en utilisant les règles vues précédemment pour arriver à un ensemble de clauses simples. On constitue ensuite des sous-ensembles de tuples de la relation pour lesquels les clauses réécrites sont vérifiées : ces tuples représentent la réponse à la question posée. Voici quelques exemples de questions exprimées en utilisant ce formalisme. On obtient ainsi des sortes de « patrons » de recherche permettant de caractériser les tuples qui sont solutions.

Dans l'exemple général (voir section 3.1), le schéma de la relation 'voiture' est voiture(NumVoit, Marque, Type, Couleur) et le schéma de la relation 'vente' est vente(DateVent, Prix, NumAch, NumVoit).

L'expression « afficher la liste des marques et des types de voitures », ou projection des champs 'Marques' et 'Types 'de la relation 'voiture', peut s'écrire sous la forme :

```
{ (m,t) | voiture(_,m,t,_) }
```

Cela signifie que les valeurs des champs 'Marque' et 'Type' des tuples de la relation 'voiture' sont représentées par les variables 'm' et 't'. Le signe '_' représente n'importe quelle valeur des autres champs 'NumVoit' et de 'Couleur'.

L'expression « afficher le type des voitures de couleur rouge », qui est une projection et une sélection sur la relation 'voiture', peut s'écrire sous la forme :

```
{ (t) | voiture(_,_,t,'rouge') }
```

Cette fois, on utilise une constante, 'rouge', dans l'expression pour fixer la valeur d'un champ et la variable, 't', pour les valeurs du champ recherchées.

On peut exprimer l'appartenance d'un champ à un ensemble par un critère. L'expression « afficher les prix de vente supérieurs à 10 000 », qui est une projection et une sélection sur la relation 'vente', peut s'écrire sous la forme :

```
{ (p) | vente(_,p,_,_) ∧ p > 10000 }
```

Enfin, la jointure entre deux relations est également possible. L'expression « afficher le type des voitures vendues et leur prix », qui met en jeu deux relations, peut s'écrire sous la forme :

```
{ (t,p) | ∃ nv voiture(nv,_,t,_) ∨ vente(_,p,_ ,nv)}
```

Il suffit d'utiliser la même variable dans les deux relations : ici 'nv', pour le champ 'NumVoit' qui sert à effectuer le lien. Le quantificateur '∃' permettra de ne sélectionner que les tuples dans les deux relations pour lesquels les valeurs de 'NumVoit' sont identiques, ce qui est exactement la définition d'une jointure (équijointure).

Le formalisme de la logique du premier ordre permet d'exprimer toutes les opérations relationnelles vues précédemment : c'est normal puisqu'il s'agit de la base de l'approche relationnelle. On peut alors considérer un SGBD comme un outil de démonstration de théorèmes, tels que l'on peut en rencontrer en intelligence artificielle, qui agirait sur un ensemble très restreint de règles.

5.3 Interrogation par QBE (*Query By Example*)

Lors de la phase de développement du prototype de SGBD basé sur l'approche relationnelle dans les laboratoires d'IBM, un premier sous-produit a été conçu : le langage d'interrogation SEQUEL, puis SQL. Une autre méthode d'interrogation a été développée à cette occasion : les **QBE** (*Query By Example*). L'idée est de disposer d'un mode d'interrogation d'une base de données sans connaître de langage et en utilisant le formalisme vu à la section précédente, présenté sous une forme graphique. Comme son nom l'indique, on va utiliser des exemples pour définir les questions. La relation est représentée sous la forme d'un tableau :

La projection d'un champ se fait en cochant la case correspondant au nom de la colonne (voir figure 3.31).

Figure 3.31

Projection du champ 'Marque' dans un QBE.

Afficher la liste des marques et des types de voitures :

```
{ (m,t) | voiture(_,m,t,_) }
```

☐ NumVoit	☒ Marque	☒ Type	☐ Couleur

- Les critères de sélection se font par des valeurs exemples (voir figures 3.32 et 3.33).

Figure 3.32 **Afficher le type des voitures de couleur rouge :**

Sélection sur un contenu et projection dans un QBE.

```
{ (t) | voiture(_,_,t,'rouge') }
```

☐ NumVoit	☐ Marque	☒ Type	☐ Couleur Rouge

Figure 3.33 **Afficher les prix de vente supérieurs à 10 000 :**

Sélection sur un critère et projection dans un QBE.

```
{ (p) | vente(_,p,_,_) ∧ p > 10000 }
```

☐ DateVent	☒ Prix	☐ NumAch	☐ NumVoit
	> 10 000		

- Les conditions situées sur la même ligne sont reliées par un « et » (voir figure 3.34).

Figure 3.34 **Afficher les prix de vente supérieurs à 10 000 et dont la date de vente a eu lieu après le 1er janvier 1997 :**

Sélection multicritère obligatoire et projection dans un QBE.

```
{ (p) | vente(d,p,_,_) ∧ (p > 10000 ∧ d > '01/01/1997')}
```

☐ DateVent	☒ Prix	☐ NumAch	☐ NumVoit
> '01/01/1997'	> 10 000		

- Les conditions situées sur deux lignes différentes sont reliées par un « ou » (voir figure 3.35).

Figure 3.35 **Afficher les prix de vente supérieurs à 10 000 ou dont la date de vente a eu lieu après le 1er janvier 1997 :**

Sélection multicritère optionnel et projection dans un QBE.

```
{ (p) | vente(d,p,_,_) ∧ (p > 10000 ∨ d > '01/01/1997')}
```

☐ DateVent	☒ Prix	☐ NumAch	☐ NumVoit
> '01/01/1997'	> 10 000		

- La jointure entre deux relations se fait en utilisant une variable, exactement comme dans une formule de la logique de premier ordre (voir figure 3.36).

Figure 3.36 **Afficher le type des voitures vendues et leur prix :**

Jointure et projection dans un QBE.

```
{ (t,p) | ∃ nv voiture(nv,_,t,_) ∨ vente(_,p,_ ,nv)}
```

☐ NumVoit	☐ Marque	☒ Type	☐ Couleur
Nv			

☐ DateVent	☒ Prix	☐ NumAch	☐ NumVoit
			Nv

Cette méthode d'interrogation simple et efficace est encore proposée par de nombreux SGBD.

Résumé

L'approche relationnelle modélise les faits de la vie réelle par des tuples, qui sont des ensembles de valeurs de différents champs (ou attributs) : (réfrigérateur, 2003, rouge) est un tuple qui représente des valeurs des champs 'Objet', 'Année', 'Couleur' liées dans le monde réel. L'ensemble des tuples s'appelle une relation. Il s'agit du concept de base qui sera manipulé par l'approche relationnelle et peut être représenté sous la forme d'une table.

Pour identifier un tuple, on utilise le contenu d'un ou de plusieurs champs que l'on nommera la **clé** d'une relation. Cette dernière est établie en utilisant le concept de dépendance fonctionnelle entre les différents champs. La clé est constituée par le plus petit ensemble de champs dont dépendent fonctionnellement les autres champs. Si plusieurs clés sont possibles, on parle de clés candidates. La clé choisie sera nommée la clé primaire.

Les opérations de manipulation de ces relations peuvent être regroupées en trois catégories :

- **Les opérations du monde ensembliste.** Produit cartésien, intersection, union et différence.
- **Les opérations spécifiques relationnelles.** Projection, sélection (ou restriction) et jointure. On a présenté également la jointure externe qui permet de répondre à des questions spécifiques même s'il ne s'agit pas d'une opération de base du relationnel.
- **Les opérations et les fonctions de calcul.** Elles ne constituent pas réellement des opérations du monde relationnel mais sont utiles pour effectuer des calculs sans recourir à un langage de programmation.

La cohérence des données contenues dans les relations est améliorée par la définition de contraintes que l'on appliquera sur les relations au moment de leur création. Ces contraintes expriment essentiellement les conditions d'appartenance à un ensemble que l'on nomme le domaine du champ. On peut décrire cet ensemble de plusieurs manières :

- Une liste exhaustive de valeurs, comme celles des jours de la semaine : « lundi », « mardi », etc.
- Le respect de propriétés, comme : « l'âge doit être compris entre 7 et 77 ans ».
- L'utilisation des valeurs d'un champ comprises dans une autre relation de référence. On parle de clé étrangère.

La mise en œuvre de ces contraintes est assurée par le SGBD en utilisant le *Langage de Définition de Données* (LDD). Après avoir présenté les concepts de relation (ou de table) et les outils qui permettent de les manipuler, on a décrit les règles qui conduisent du modèle conceptuel présenté au chapitre précédent à un ensemble de relations constituant le modèle logique de la base de données. Les liaisons entre les relations, qui expriment les liens de sens dans la réalité, seront établies dynamiquement par l'opération fondamentale de l'approche relationnelle qui est la jointure.

On évalue ensuite la qualité de ces relations en vérifiant leur conformité par rapport à des propriétés que l'on appelle les formes normales. Ces propriétés visent essentiellement à détecter la redondance et la cohérence des données dans les relations. On a présenté dans ce chapitre les quatre formes normales les plus courantes :

- la première forme normale qui interdit les champs multivalués ;
- la deuxième forme normale qui détecte une relation de dépendance entre une partie de la clé et un champ ;

- la troisième forme normale qui détecte une dépendance « transitive » entre une clé et un champ, c'est-à-dire qu'un champ non-clé dépend d'un autre champ non-clé qui dépend lui même de la clé ;
- la forme normale de Boyce-Codd qui détecte la relation de dépendance entre un champ non-clé et une partie d'une clé ; c'est une extension de la troisième forme normale qui est plus restrictive.

La normalisation est une méthode de réorganisation qui consiste à décomposer une relation pour la rendre conforme aux formes normales. La recomposition des données se fait alors par une opération de jointure qui peut se révéler coûteuse en ressources. C'est pour cette raison de performance que certaines relations sont parfois laissées en forme non normalisée. Les deux étapes précédentes de passage du modèle conceptuel au schéma relationnel et de normalisation peuvent être quasiment automatisées. Enfin, on a abordé les bases du fondement logique de l'approche relationnelle. L'objectif est de présenter une méthode d'interrogation intuitive et graphique d'une base de données qui en découle : les QBE ou « interrogation par l'exemple ».