2 Modèle logique de données (MLD)

Maintenant que le MCD est établi, on peut le traduire en différents systèmes logiques et notamment les bases de données relationnelles qui proposent une vision plus concrète pour modéliser la situation.

2.1 Systèmes logiques

Avant l'apparition des systèmes de gestion de base de données (SGBD ou DBMS pour Data Base Management System), les données étaient stockées dans des fichiers binaires et gérées par des programmes exécutables (développés en Basic, Cobol ou Dbase, par exemple). [Gabay] propose à ce sujet une traduction d'un MPD vers un MLD fichier. Mais la maintenance des programmes (en cas de modification de la structure des données, notamment) était très problématique.

Sont alors apparus les SGBD hiérarchiques dans lesquels les données sont organisées en arbre (IMS-DL1 d'IBM, par exemple), puis les SGBD réseaux dans lesquels les données sont organisées selon un graphe plus général (IDS2 de Bull, par exemple). [Matheron, Nanci *et al.*, Gabay] décrivent la traduction d'un MPD vers un MLD Codasyl (base de données réseaux). Ces deux types de SGBD sont dit navigationnels car on peut retrouver l'information à condition d'en connaître le chemin d'accès.

Aujourd'hui, ils sont largement remplacés par les SGBD relationnels (SGBDR) avec lesquels l'information peut être obtenue par une requête formulée dans un langage quasiment naturel (la langage SQL pour Structured Query Langage). Parmi les SGBDR les plus répandus nous trouvons Oracle, SQL Server et DB2. Nous nous contentons ici d'exposer le modèle logique de données relationnel (MLDR).

Plus récemment, sont apparus le modèle logique orienté objet et même des SGBD orientés objets. Pourtant, les SGBD relationnels restent extrêmement majoritaires, tandis que l'approche orienté objet est parfaitement adaptée au développement d'applications clientes dynamiques et liées aux données du système d'information.

2.2 Modèle logique relationnel

Concentrons-nous désormais sur le MLDR.

2.2.1 Tables, lignes et colonnes

Lorsque des données ont la même structure (comme par exemple, les renseignements relatifs aux clients), on peut les organiser en table dans laquelle les colonnes décrivent les champs en commun et les lignes contiennent les valeurs de ces champs pour chaque enregistrement (tableau 3).

numéro client	nom	prénom	adresse
1	Dupont	Michel	127, rue...
2	Durand	Jean	314, boulevard...
3	Dubois	Claire	51, avenue...
4	Dupuis	Marie	2, impasse...
...	...	...	...

TAB. 3 – *Contenu de la table* `clients`*, avec en première ligne les intitulés des colonnes*

2.2.2 Clés primaires et clés étrangères

Les lignes d'une table doivent être uniques, cela signifie qu'une colonne (au moins) doit servir à les identifier. Il s'agit de la clé primaire de la table.

L'absence de valeur dans une clé primaire ne doit pas être autorisée. Autrement dit, la valeur vide (`NULL`) est interdite dans une colonne qui sert de clé primaire, ce qui n'est pas forcément le cas des autres colonnes, dont certaines peuvent ne pas être renseignées à toutes les lignes.

De plus, la valeur de la clé primaire d'une ligne ne devrait pas, en principe, changer au cours du temps.

Par ailleurs, il se peut qu'une colonne `Colonne1` d'une table ne doive contenir que des valeurs prises par la colonne `Colonne2` d'une autre table (par exemple, le numéro du client sur une commande doit correspondre à un vrai numéro de client). La `Colonne2` doit être sans doublons (bien souvent il s'agit d'une clé primaire). On dit alors que la `Colonne1` est clé étrangère et qu'elle référence la `Colonne2`.

Par convention, on souligne les clés primaires et on fait précéder les clés étrangères d'un dièse `#` dans la description des colonnes d'une table :

```
clients(numéro client, nom client, prénom, adresse client)
commandes(numéro commande, date de commande, #numéro client (non vide))
```

Remarques :
- une même table peut avoir plusieurs clés étrangères mais une seule clé primaire (éventuellement composées de plusieurs colonnes) ;
- une colonne clé étrangère peut aussi être primaire (dans la même table) ;
- une clé étrangère peut être composée (c'est le cas si la clé primaire référencée est composée) ;
- implicitement, chaque colonne qui compose une clé primaire ne peut pas recevoir la valeur vide (`NULL` interdit) ;
- par contre, si une colonne clé étrangère ne doit pas recevoir la valeur vide, alors il faut le préciser dans la description des colonnes.

Les SGBDR vérifient au coup par coup que chaque clé étrangère ne prend pas de valeurs en dehors de celles déjà prises par la ou les colonne(s) qu'elle référence. Ce mécanisme qui agit lors de l'insertion, de la suppression ou de la mise à jour de lignes dans les tables, garantit ce que l'on appelle l'intégrité référentielle des données.

2.2.3 Schéma relationnel

On peut représenter les tables d'une base de données relationnelle par un schéma relationnel dans lequel les tables sont appelées relations et les liens entre les clés étrangères et leur clé primaire est symbolisé par un connecteur (figure 26).

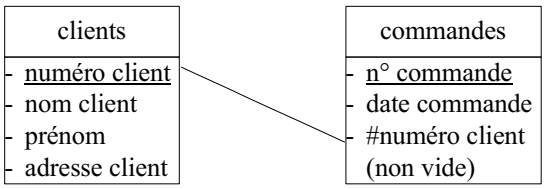

FIG. 26 – *Schéma relationnel simple entre deux tables*

Certains éditeur inscrivent sur le connecteur un symbole 1 côté clé primaire et un symbole ∞ côté clé étrangère (à condition que celle-ci ne soit pas déjà clé primaire). Il faut prendre garde avec cette convention, car le symbole ∞ se trouve du côté opposé à la cardinalité maximale `n` correspondante.

2.3 Traduction d'un MCD en un MLDR

Pour traduire un MCD en un MLDR, il suffit d'appliquer cinq règles.

Notations : on dit qu'une association binaire (entre deux entités ou réflexive) est de type :
- 1 : 1 (un à un) si aucune des deux cardinalités maximales n'est `n` ;
- 1 : n (un à plusieurs) si une des deux cardinalités maximales est `n` ;
- n : m (plusieurs à plusieurs) si les deux cardinalités maximales sont `n`.

En fait, un schéma relationnel ne peut faire la différence entre 0,n et 1,n. Par contre, il peut la faire entre 0,1 et 1,1 (règles 2 et 4).

Règle 1 : toute entité devient une table dans laquelle les attributs deviennent les colonnes. L'identifiant de l'entité constitue alors la clé primaire de la table.

Par exemple, l'entité `articles` de la figure 13 devient la table :

`articles(`<u>`numéro article`</u>`, désignation, prix unitaire de vente)`

Règle 2 : une association binaire de type 1 : n disparaît, au profit d'une clé étrangère dans la table côté 0,1 ou 1,1 qui référence la clé primaire de l'autre table. Cette clé étrangère ne peut pas recevoir la valeur vide si la cardinalité est 1,1.

Par exemple, l'association `livrer` de la figure 13 est traduite par :

`fournisseurs(`<u>`n° fournisseur`</u>`, nom contact, n° téléphone contact)`
`livraisons(`<u>`n° livraison`</u>`, date de livraison, nom livreur, #n° fournisseur (non vide))`

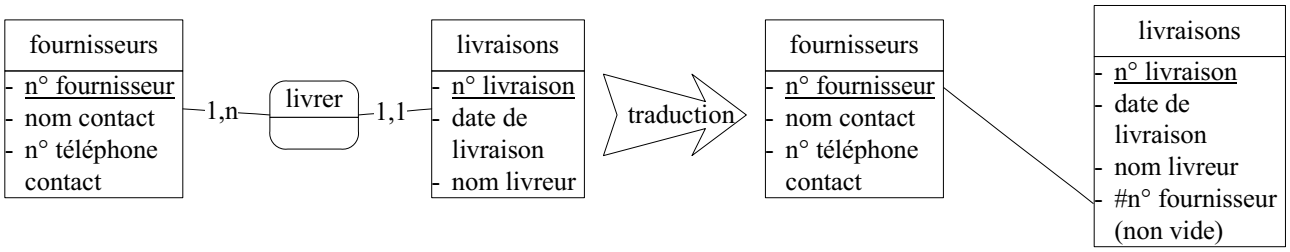

FIG. 27 – *Traduction d'une association de type 1 : n*

Il ne devrait pas y avoir d'attribut dans une association de type 1 : n, mais s'il en reste, alors ils glissent vers la table côté 1.

Règle 3 : une association binaire de type n : m devient une table supplémentaire (parfois appelée table de jonction, table de jointure ou table d'association) dont la clé primaire est composée de deux clés étrangères (qui référencent les deux clés primaires des deux tables en association). Les attributs de l'association deviennent des colonnes de cette nouvelle table.

Par exemple, l'association `concerner (1)` de la figure 13 est traduite par la table supplémentaire `lignes de commande` :

```
lignes de commande(#n° commande, #n° article, quantité commandée)
```

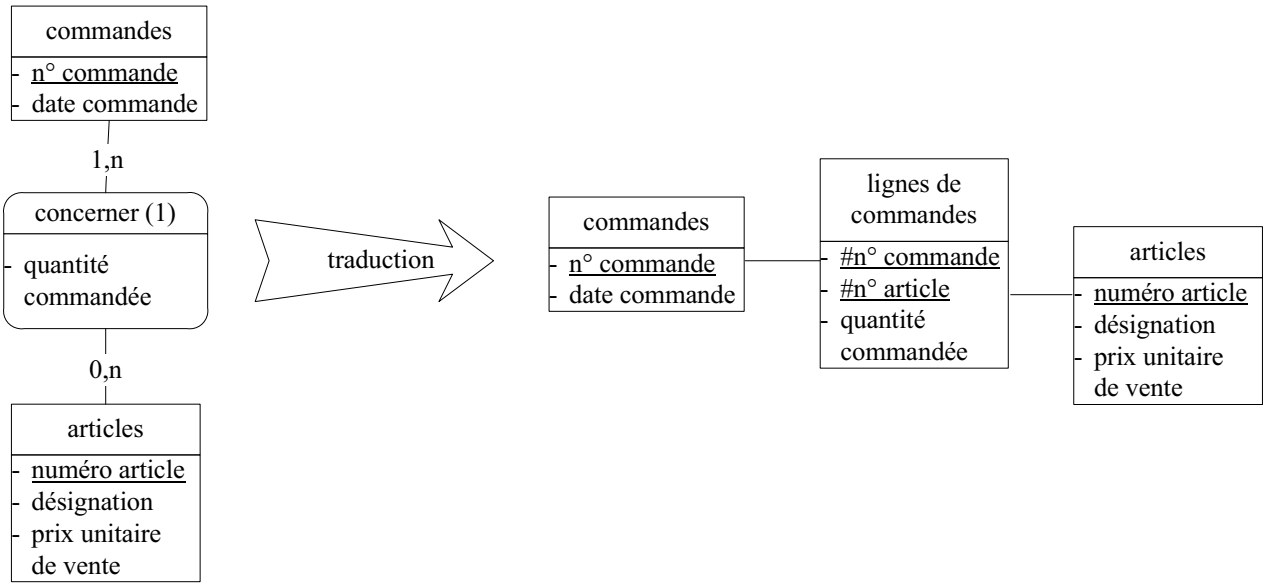

FIG. 28 – *Traduction d'une association de type n : m*

Règle 4 : une association binaire de type 1 : 1 est traduite comme une association binaire de type 1 : n sauf que la clé étrangère se voit imposer une contrainte d'unicité en plus d'une éventuelle contrainte de non vacuité (cette contrainte d'unicité impose à la colonne correspondante de ne prendre que des valeurs distinctes).

Si les associations fantômes ont été éliminées, il devrait y avoir au moins un côté de cardinalité 0,1. C'est alors dans la table du côté opposé que doit aller la clé étrangère. Si les deux côtés sont de cardinalité 0,1 alors la clé étrangère peut être placée indifféremment dans l'une des deux tables.

Par exemple, l'association `diriger` de la figure 29 est traduite par :

```
services(n° service, nom service, #numéro employé (non vide, unique))
employés(numéro employés, nom)
```

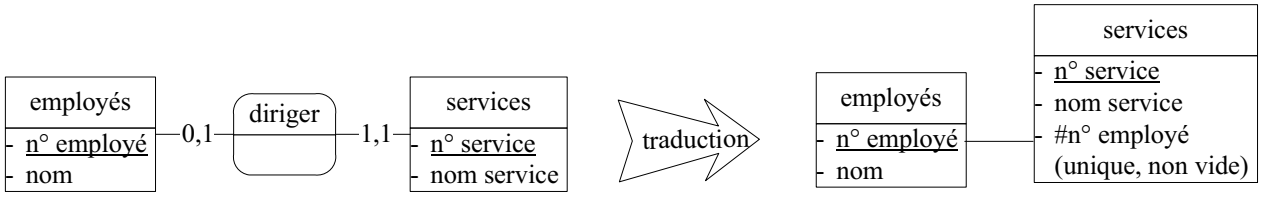

FIG. 29 – *Traduction d'une association de type 1 : 1*

En réalité, la règle 4 proposée ici considère qu'une association binaire de type 1 : 1 correspond à une association binaire de type 1 : n particulière. Une alternative consiste à voir une association binaire de type 1 : 1 comme une association binaire de type n : m particulière. Il suffit pour cela d'ajouter une contrainte d'unicité sur chacune des clés étrangères de la table de jonction supplémentaire :

```
services(n° service, nom service)
directions(#n° service (unique), #numéro employé (unique)
employés(numéro employés, nom)
```

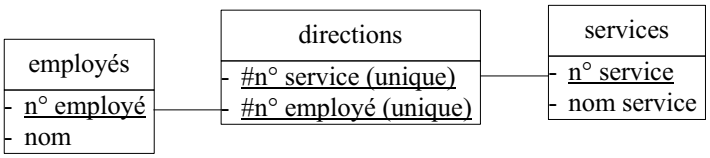

FIG. 30 – *Traduction alternative d'une association de type 1 : 1*

Mais rien ne garantit, dans cette traduction alternative (figure 30), qu'un service possède un dirigeant, alors que c'est obligatoire. La première traduction (figure 29) est donc préférable.

Remarque : d'autres techniques sont parfois proposées pour cette règle 4 (fusionner les tables, utiliser une clé primaire identique, utiliser deux clés étrangères réflexives) mais elles ne sont pas exploitables dans le cas général.

Règle 5 : une association non binaire est traduite par une table supplémentaire dont la clé primaire est composée d'autant de clés étrangères que d'entités en association. Les attributs de l'association deviennent des colonnes de cette nouvelle table.

Par exemple, l'association `projeter` de la figure 8 devient la table :

```
projections(#n° film, #n° salle, #n° créneau, tarif)
```

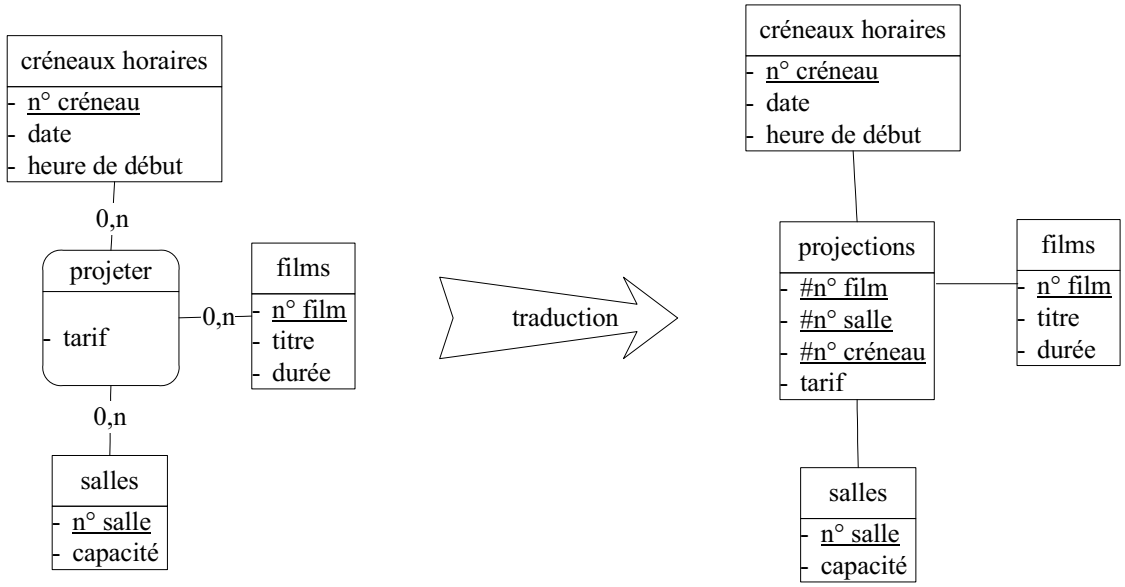

FIG. 31 – *Traduction d'une association ternaire*