Informatique

Cours

SQL- Modèles entité-association et relationnel

Ipesup

Objectifs

À l'issue de cette section, l'étudiant devra être capable de :

- comprendre le principe du modèle entité–association et identifier les entités, leurs attributs et les associations entre entités ;
- traduire un diagramme entité–association en un schéma relationnel ;
- comprendre la notion de clé primaire et son rôle pour identifier de façon unique les lignes d'une table ;
- utiliser des clés étrangères pour représenter des associations entre relations ;
- modéliser les associations $1 : N$ et $* : *$ à l'aide de tables de liaison et gérer les attributs à valeurs multiples.

La gestion de données est l'une des tâches les plus courantes en informatique. À haut niveau, la gestion de données recouvre un ensemble d'opérations variées, parmi lesquelles :

- l'acquisition des données (par exemple des capteurs météos, des informations de trafic ferroviaire, la saisie des notes des élèves pour une matière, etc.) ;
- la validation et la mise en forme des données (par exemple, la normalisation de données issues de capteurs, la suppression de valeur impropres, etc.) ;
- le stockage et l'organisation des données sur un support physique (format des fichiers de stockage, organisation de ces derniers sur les disques durs, etc.) ;
- l'extraction et l'interrogation des données (pour des données météos, répondre à des questions comme « quelle température faisait-il le 2 janvier 2022 ? », « quel est le niveau de précipitations cumulé sur le mois de février 2022 ? ») ;
- le contrôle d'accès aux données (garantir que seul le patient et son médecin ont accès au dossier médical du patient, et pas n'importe quel professionnel de santé).

Malgré la très grande variété des types de données, des domaines d'activité et types de traitements, il existe une approche systématique que l'on peut appliquer aussi bien à des données météorologiques qu'à des ensembles d'élèves et à leurs notes, ou aux produits d'un site de commerce en ligne. Cette approche est celle du *modèle relationnel*.

Le modèle relationnel a été introduit en 1969 par l'informaticien britannique Edgar Frank Codd. À cette époque aucune solution systématique n'existe. Chaque programme, chaque système commercial représente les données d'une façon ad-hoc, et propose diverses manières d'interroger et mettre à jour ces dernières. C'est dans ce contexte que Codd s'intéresse à la question de la *redondance des données* (le fait que certaines informations soient dupliquées, soit volontairement pour accroître les performances, soit involontairement suite à une mauvaise conception du système).

Codd souhaite discuter de ces problèmes dans un cadre rigoureux. Il a donc l'idée de modéliser les données de la façon la plus simple possible, avec le minimum de structure : des ensembles de n-uplets de valeurs scalaires. C'est le modèle relationnel. À ce modèle de données, Codd ajoute des opérations spécifiques, issues de la théorie des ensembles et de la logique du premier ordre : l'algèbre relationnelle. Par sa simplicité et son élégance, le modèle relationnel devient

rapidement le formalisme sur lequel se construit alors la théorie des bases de données. Ce modèle est si populaire qu'il est repris au début des années 1970 par Donald D. Chamberlain et Raymond F. Boyce comme base pour une implémentation pratique d'un langage de requêtes : le *Structured Query Language* (SQL).

Dans la suite, nous commençons par introduire le modèle *entité-association*, formalisme graphique de haut niveau permettant de spécifier simplement des données en dehors de toute considération d'implémentation propre au modèle relationnel et au langage SQL. Nous présentons ensuite le modèle relationnel en tant que tel, avant d'aborder le langage SQL.

11.1 Le modèle entité-association

Supposons que nous souhaitions représenter une base de données de films. Plus précisément, nous souhaitons stocker pour chaque film :

— son titre (en français) ;
— son année de sortie ;
— sa durée en minutes ;
— ses genres (tel que « comédie », « drame », « animation », etc.) ;
— le ou les pays de production ;
— la liste de son ou ses réalisateurs ;
— la liste de ses acteurs, avec leurs rôles dans le film ;
— le titre original du film (si différent du titre en français) ;
— si le film a remporté l'Oscar du meilleur film ainsi que le nom de la personne ayant présenté la cérémonie et l'année de cette dernière.

Cette base de données doit permettre ensuite de répondre à des requêtes telles que :

— Dans quels films a joué l'acteur Clint Eastwood ?
— Qui a réalisé le film *Le bon, la Brute et le Truand* ?
— Quelle est la durée moyenne d'un film d'animation ?
— Quels réalisateurs ont joués dans leur propres films ?
— Pour chaque année de production, quel est le film le plus long sorti cette année-là ?
— etc.

En regardant la liste informelle des caractéristiques de nos films, on remarque que certaines données sont « composites » (par exemple un film) alors que d'autres semblent plus primitives (une année, une durée). On remarque aussi que des objets complexes peuvent être mis en relation (par exemple, une personne joue dans un film particulier).

Une façon de spécifier une telle base de données est d'utiliser le modèle *entité-association*.

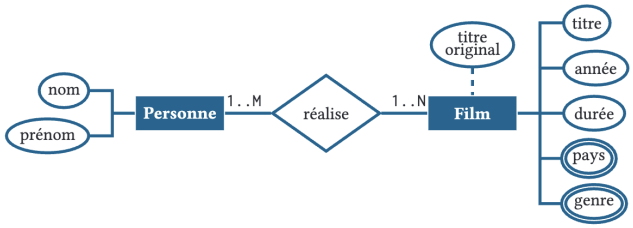
Ce modèle consiste en un langage graphique permettant d'organiser visuellement les données principales d'une base de données, leurs propriétés et la façon dont elles sont reliées.

Dans ce modèle, une *entité* est un objet d'étude ayant une existence propre et pouvant être identifié de façon unique. Dans notre exemple, les films et les personnes (que ce soient des acteurs, réalisateurs ou présentateurs) sont des entités. Chaque entité dispose d'un ensemble d'attributs la caractérisant. Dans le modèle EA, les entités sont représentées graphiquement par des rectangles auxquels sont rattachés leurs attributs représentés par des ellipses.

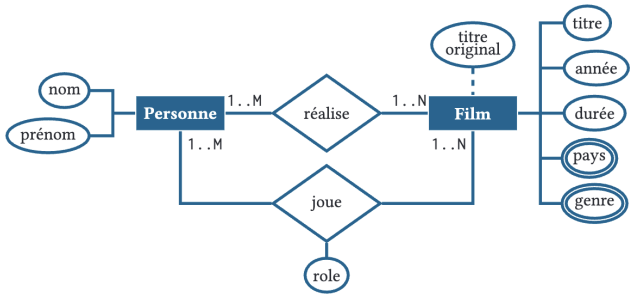
Exemple

Dans la figure ci-dessus, **Personne** et **Film** représentent des ensembles d'entités (les films et les personnes de notre base). Une personne est caractérisée par son nom et son prénom. Un film est décrit par son titre, son année et sa durée. Un film pouvant avoir plusieurs pays et plusieurs genres, ces derniers sont des attributs à valeurs multiples. Ils sont dénotés par des ellipses à double bordure. Enfin, le titre original du film peut être absent (dans le cas d'un film français). Le caractère optionnel de l'attribut est indiqué par le lien en pointillés.

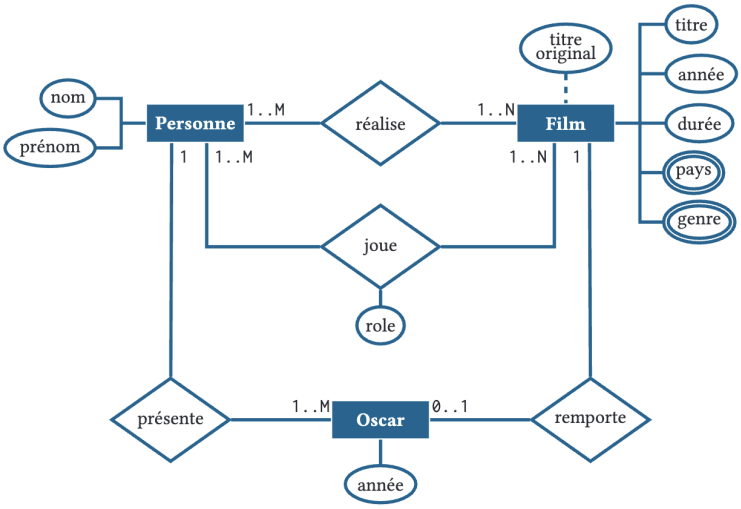
Comme on le voit, le modèle EA propose une façon visuelle de spécifier les entités et leurs attributs. Elle permet aussi de représenter les associations entre plusieurs ensembles d'entités. Par exemple, le fait qu'une personne ait réalisé un film peut être représenté par l'association *réalise* de la figure ci-dessous :



Le losange *réalise* est une association. Les indications « 1..M » et « 1..N », à chaque extrémité de l'association indiquent qu'un nombre arbitraire N de films (au moins un) peut être associé à un nombre arbitraire M de personnes (au moins une). En effet, un film peut être réalisé par plusieurs personnes. À l'inverse, une même personne peut avoir réalisé plusieurs films. Une telle association est dite de cardinalité $M : N$ (parfois aussi noté $* - *$ et appelée *many-to-many* en anglais). La notation $a..b$ signifie que chaque élément du côté opposé de l'association peut être associé à au moins a et au plus b entité. Lorsque a et b sont identiques, on indique simplement l'un des deux, *i.e.* on écrit 1 pour « 1..1 ». Tout comme les entités, les associations peuvent posséder des attributs. Par exemple, l'association *joue*, représentée ci-dessous, indique qu'une personne joue un rôle particulier dans un film donné :



Comme l'association *réalise*, l'association *joue* est de cardinalité $M : N$ (une même personne peut avoir joué dans plusieurs films, et plusieurs personnes jouent dans le même film). Une information supplémentaire est stockée avec cette association : le rôle joué par la personne dans le film. Nous terminons notre présentation informelle du modèle EA en donnant le diagramme complet de notre base de données.



Ce diagramme comporte une entité supplémentaire *Oscar* représentant l'Oscar du meilleur film pour une année donnée. La cérémonie des Oscars est toujours présentée par exactement une personne [1] et une personne peut présenter plusieurs fois la cérémonie. C'est un exemple d'association de cardinalité $1 : N$ (ou $1 - *$, *one-to-many* en anglais). De façon parfaitement symétrique, on parle parfois de relation $N : 1$ (ou $* - 1$, *many-to-one* en anglais) lorsque la cardinalité multiple se trouve à gauche. Enfin, un Oscar n'est remporté que par exactement un film et un film ne peut avoir qu'au plus un Oscar du meilleur film (nous ne représentons pas dans notre base les autres types de récompenses tels que meilleur premier rôle, meilleure bande originale, etc.). C'est un cas d'association $1 : 1$ (*one-to-one* en anglais). Le caractère optionnel peut être indiqué par 0..1 sur le diagramme.

En conclusion, le modèle EA est un outil utile lors de la phase initiale de conception d'une base de données. Il permet de spécifier, à un niveau abstrait, les différentes entités que l'on souhaite manipuler et de se poser la question des associations entre ces dernières. Une attention particulière doit être apportée à la cardinalité des associations. En effet, si celles-ci sont bien spécifiées, leur encodage dans une base de données relationnelle peut alors être faite de façon systématique en appliquant quelques règles de base. Avant de donner ces règles, nous présentons le modèle relationnel et ses spécificités.

11.2 Le modèle relationnel

11.2.1 Relation

Dans ce modèle, comme dans le modèle EA, les objets complexes (comme ici nos films) sont appelés des *entités* ou des *enregistrements*. Le modèle relationnel permet de définir des ensembles d'entités appelés *relations*. Une relation est un ensemble de n-uplets, tous de même

1. Nous faisons cette simplification pour les besoins de l'exemple.

taille. Chaque composante d'un n-uplet est appelé un *attribut*. Les attributs sont caractérisés par leur nom et leur *domaine* (aussi appelé *type*). Le nom de la relation associée aux noms et domaines de tous les attributs est appelé le *schéma* de la relation. Le nombre d'attributs est appelé le *degré* ou l'*arité* de la relation. Le nombre d'éléments qu'elle contient est appelé le *cardinal* de la relation.

Commençons par modéliser la relation *Film*. Dans un premier temps, nous ignorons le fait que le genre et le pays sont des attributs multiples et supposons qu'un film possède exactement un genre et un pays. Un schéma possible pour cette relation *Film* est donné ci-dessous, avec des valeurs pour les entités qu'elle contient :

$$Film(\text{titre} : text,\ \text{année} : int,\ \text{durée} : int,\ \text{genre} : text,\ \text{pays} : text)$$

$$Film = \begin{cases} (\text{"La mélodie du bonheur"},\ 1965,\ 174,\ \text{"musical"},\ \text{"États-Unis"}) \\ (\text{"Crocodile Dundee 2"},\ 1988,\ 112,\ \text{"aventure"},\ \text{"Australie"}) \\ (\text{"Le livre de la jungle"},\ 1967,\ 78,\ \text{"animation"},\ \text{"États-Unis"}) \\ (\text{"Casino Royale"},\ 2006,\ 144,\ \text{"espionnage"},\ \text{"Royaume-Uni"}) \\ (\text{"Léon"},\ 1994,\ 110,\ \text{"drame"},\ \text{"France"}) \\ \vdots \end{cases}$$

Il est courant d'utiliser une représentation tabulaire pour les relations :

Film

titre `text`	***année*** `int`	***durée*** `int`	***genre*** `text`	***pays*** `text`
La mélodie du bonheur	1965	174	musical	États-unis
Crocodile Dundee 2	1988	112	aventure	Australie
Le livre de la jungle	1967	78	animation	États-unis
Casino Royale	2006	144	espionnage	Royaume-uni
Léon	1994	110	drame	France
⋮				

Ainsi, on parle souvent de *table* plutôt que de relation, de *ligne* plutôt que d'entité et de *colonne* plutôt que d'attribut, la représentation tabulaire montrant naturellement la correspondance entre ces concepts.

Le modèle relationnel tel que défini par Codd ne donne pas de liste précise des domaines des attributs. Il suppose juste l'existence d'un certain nombre de domaines dans lesquels les attributs prennent leurs valeurs et d'opérations prédéfinies sur ces domaines. Dans le cadre de ce chapitre, on supposera l'existence de trois domaines :

— `int` : les entiers signés d'une taille fixe (non spécifiée) ;
— `text` : les chaînes de caractères d'une taille maximale fixe (non spécifiée) ;
— `float` : les nombres flottants d'une taille fixe (non spécifiée).

Nous détaillerons dans la suite les opérations définies sur ces domaines. Le modèle relationnel définit de plus une valeur spéciale notée `NULL`, valide pour tous les types, et qui représente une absence de valeur. Cette dernière joue un rôle semblable au pointeur `NULL` du langage C, mais possède un comportement bien particulier (notamment dans les opérations booléennes et les comparaisons) que nous expliquerons au fur et à mesure.

La relation *Film* précédemment donnée est une première étape dans notre modélisation relationnelle des films. Il manque cependant crucialement l'information des acteurs, de leur rôle dans chaque film et des réalisateurs. De même, nous avons imposé une simplification (unicité du pays et du genre) que nous souhaitons lever maintenant.

Dans ce but, nous pourrions être tentés de rajouter à notre relation *Film* des colonnes supplémentaires pour indiquer les acteurs et réalisateurs des films ainsi que les genres et les pays. Cette approche montre vite ses limites. En effet, le nombre d'acteurs d'un film étant variable (de même que le nombre de réalisateurs), on se retrouverait à rajouter des colonnes telles que :

...	na_1 text	pa_1 text	ra_1 text	...	na_m text	pa_m text	ra_m text	nr_1 text	pr_1 text	...	nr_n text	pr_n text
					⋮							

où les na_i, pa_i, ra_i sont les noms, prénoms et rôles des acteurs et les nr_j, pr_j sont les noms et prénoms des réalisateurs.

Une telle modélisation est inélégante :

— il faut fixer *a priori* le nombre maximal de réalisateurs et d'acteurs d'un film ;
— si un film a plus de réalisateurs ou d'acteurs que cette limite, on ne peut les stocker ;
— si un film a moins d'acteurs ou de réalisateurs que la limite, il faut tout de même remplir toutes les colonnes, avec des valeurs par défaut (par exemple `NULL`).

Nous sommes ici dans la situation où l'on veut pouvoir *associer* un nombre arbitraire d'entités (par exemple des acteurs) à une entité donnée (un film). Avant de présenter la façon de modéliser de telles associations, nous allons modéliser des personnes (par le couple de leur nom et de leur prénom). Nous verrons ensuite comment associer des personnes à un film et comment indiquer qu'elles sont associées en qualité d'acteur ou de réalisateur. Nous pourrons ensuite appliquer une méthode semblable pour les pays et les genres.

L'ensemble des personnes peut être modélisé par la table *Personne* suivante :

Personne

nom text	prénom text
Plummer	Christophe
Andrews	Julie
Reno	Jean
Craig	Daniel
Green	Eva
Besson	Luc
⋮	

Cette modélisation pose cependant un problème que nous présentons maintenant.

11.2.2 Clé primaire

Notre but est de pouvoir associer des films à des personnes (en qualité d'acteur ou de réalisateur). La modélisation $Personne(\text{nom} : text, \text{prénom} : text)$ a cependant un défaut. En effet,

il est possible que deux personnes différentes aient le même nom et le même prénom. Cette situation est courante, par exemple si on considère la liste des étudiants d'une université, des électeurs d'une grande ville, etc. Mais même sur l'échantillon relativement réduit que représentent les acteurs, le problème se produit. En effet, l'ancien basketteur Américain Michael Jordan joue dans le film *Space Jam*. Mais il est homonyme de l'acteur Michael Jordan qui joue dans *Black Panther*. Le nom et le prénom ne permettent donc pas d'identifier de façon unique une personne. Nous devons donc trouver un moyen de différencier ces deux entités.

Le modèle relationnel définit la notion de *clé primaire*. Une clé est un sous-ensemble des attributs d'une table qui identifie une ligne de façon unique. Une clé primaire est une clé particulière identifiée comme clé pour la relation donnée. Pour la relation *Film*, l'attribut *titre* ne constitue pas une clé, car il est possible que plusieurs films portent le même titre. Par contre, on peut raisonnablement penser que le triplet $(titre, année, durée)$ identifie un film de façon unique (car il semble improbable que la même année, deux films différents ayant le même titre et la même durée à la minute près soient produits). Lorsque l'on donne le schéma d'une relation, la convention veut que l'on souligne l'ensemble des attributs constituant la clé primaire. On a donc pour la relation *Film* :

$$Film(\underline{\text{titre}} : text,\ \underline{\text{année}} : int,\ \underline{\text{durée}} : int,\ \text{genre} : text,\ \text{pays} : text)$$

On ne peut cependant trouver un tel sous-ensemble d'attributs pour la relation *Personne*. C'est une violation du modèle relationnel qui impose que chaque entité (chaque ligne) d'une table soit identifiable de façon unique par une clé primaire. Une façon de faire dans ce cas consiste à ajouter à chaque entité un identifiant unique. Pour la relation *Personne*, cela donne

$$Personne(\underline{\text{pid}} : int,\ \text{nom} : text,\ \text{prénom} : text)$$

Ici, nous avons artificiellement ajouté un attribut de type entier, qui identifie chaque ligne de la table. C'est le rôle du processus qui remplit la table d'associer un tel identifiant unique pour chaque entité. Il est considéré comme une bonne pratique d'utiliser un identifiant unique plutôt que de supposer que des données de la vie réelle vont être uniques. Ainsi, on peut aussi redéfinir la relation *Film* comme ceci :

$$Film(\underline{\text{fid}} : int,\ \text{titre} : text,\ \text{année} : int,\ \text{durée} : int,\ \text{genre} : text,\ \text{pays} : text)$$

Attention, une clé primaire *ne peut jamais valoir* `NULL`. Nos entités étant maintenant identifiées de façon unique, nous pouvons nous attaquer au problème d'associer plusieurs entités entre elles.

11.2.3 Clé étrangère

Nous nous penchons maintenant sur la méthode permettant de relier des relations entre elles. Nous avons pour cela le diagramme EA comme guide. Considérons dans un premier temps le cas d'une association $1-1$ telle que *remporte*. Cette dernière associe chaque *Film* à son *Oscar* (s'il en a remporté un). On peut modéliser cette association en étendant la table *Film* :

$$Film(\text{fid} : int,\ \text{titre} : text,\ \text{année} : int,\ \text{durée} : int,\ \text{oscar} : int)$$

Nous omettons volontairement le pays et le genre dans cette nouvelle modélisation de la table *Film* ; ils seront ajoutés ultérieurement. Comme on le voit, pour une relation $1-1$, rien n'est plus simple que de mettre une nouvelle colonne du type approprié. On choisit ici un entier pour représenter l'année d'obtention de l'Oscar (qui est usuellement l'année suivant l'année de sortie du film, mais pas toujours). On pourra utiliser `NULL` pour indiquer que le film n'a pas remporté d'Oscar. Par exemple :

Film

fid int	*titre* text	*année* int	*durée* int	*oscar* int
42	La mélodie du bonheur	1965	174	1966
38	Crocodile Dundee 2	1988	112	NULL
14	Le livre de la jungle	1967	78	NULL
499	Casino Royale	2006	144	NULL
302	Léon	1994	110	NULL
771	The Artist	2011	100	2012
⋮				

Les identifiants des films (colonne *fid*) sont arbitraires ; nous savons simplement qu'ils sont uniques. La situation est assez simple.

Cependant, si l'entité à associer comporte plusieurs attributs, on peut assez vite se retrouver avec des tables ayant énormément de colonnes. De plus, il est possible qu'un grand nombre de colonnes contiennent `NULL`. C'est le cas ici car seul un film par an peut avoir l'Oscar du meilleur film. Enfin, nous avons ici deux entités distinctes « Oscar » et « Film » de notre diagramme entité-association qui se retrouvent confondues en une seule entité (au sens du modèle relationnel), dans une table *Film*.

Une solution alternative, plus flexible, consiste à utiliser la notion de *clé étrangère*. Une clé étrangère est un ensemble d'attributs d'une table qui forme une clé primaire dans une autre table. On utilise donc deux tables distinctes pour représenter les deux entités :

$$Film(\underline{\text{fid}} : int,\ \text{titre} : text,\ \text{année} : int,\ \text{durée} : int) Oscar(\underline{\text{fid}} : int,\ \text{année} : int)$$

Dans les schémas ci-dessus, on a dénoté par un soulignement haché la colonne *fid* de la table *Oscar*. Cette dernière constitue une clé étrangère. Cette contrainte implique que la colonne *fid* de la table *Oscar* ne peut contenir que des entiers apparaissant comme clé primaire dans la table *Film*. En reprenant les valeurs de l'exemple précédent, on a donc :

Film

fid int	*titre* text	*année* int	*durée* int
42	La mélodie du bonheur	1965	174
38	Crocodile Dundee 2	1988	112
14	Le livre de la jungle	1967	78
499	Casino Royale	2006	144
302	Léon	1994	110
771	The Artist	2011	100
⋮			

Oscar

fid int	*année* int
771	2012
42	1966
⋮	

Intuitivement, la colonne *fid* de la table *Oscar* peut être interprétée comme un « pointeur » vers l'unique ligne de la table *Film* ayant la même valeur dans la colonne *fid*. L'unicité est garantie par le fait que *fid* dans *Film* est une clé primaire. Cette modélisation relationnelle correspond bien à la représentation en diagramme entité-association.

L'utilisation d'une clé étrangère pour « pointer » entre deux tables peut naturellement être appliquée au cas des relations $1-*$. Si on considère maintenant les présentateurs d'Oscar (on se souvient qu'une personne peut présenter plusieurs fois les Oscars mais qu'un Oscar n'est présenté que par une personne), la table *Oscar* devient :

$$Oscar(\underline{\text{fid}} : int,\ \text{année} : int,\ \text{pres\_id} : int)$$

Ici, l'attribut *pres_id* est une clé étrangère représentant l'identifiant de la personne. En d'autres termes, la valeur de *pres_id* doit être l'un des *pid* de la table *Personne*. Ainsi, et de façon systématique, dans le cas d'une association $1-N$, on peut ajouter à la relation se trouvant « du côté N » une clé étrangère vers la clé primaire de l'entité se trouvant « du côté 1 » de l'association.

Le cas des associations $*-*$ est plus complexe. En effet, si on voulait appliquer la même technique il faudrait :

— soit pouvoir associer dans la table *Film*, pour un film donné, un nombre arbitraire de clés primaires de personnes (qui ont réalisé ce film) ;
— soit de façon symétrique pouvoir associer dans la table *Personne* un nombre arbitraire de clés primaires de films (que cette personne a réalisés).

Plus complexe encore, l'association *joue* devrait d'une façon ou d'une autre réussir à stocker également le nom du rôle joué par la personne dans un film.

La solution pour représenter des associations $*-*$ consiste à passer par une table auxiliaire, appelée *table de liaison* (ou table de jonction). Cette table de liaison représente explicitement l'association $*-*$ que l'on souhaite représenter. Par exemple, dans le cadre des réalisateurs, il suffit de créer une table

$$Réalise(\text{fid} : int,\ \text{pid} : int)$$

dans laquelle *fid* est une clé étrangère vers *Film* et *pid* une clé étrangère vers *Personne*. Une ligne dans la table *Réalise* s'interprète donc naturellement comme : « la personne dont l'identifiant est *pid* réalise le film dont l'identifiant est *fid* ».

Film

fid int	*titre* text	*année* int	*durée* int
42	La mélodie du bonheur	1965	174
38	Crocodile Dundee 2	1988	112
14	Le livre de la jungle	1967	78
499	Casino Royale	2006	144
302	Léon	1994	110
771	The Artist	2011	100
⋮			

Personne

pid int	*nom* text	*prénom* text	*cid* int
17	Wise	Robert	18
18	Cornell	John	33
21	Reitherman	Wolfgang	18
42	Campbell	Martin	34
60	Besson	Luc	19
70	Hazanavicius	Michel	19
⋮			

Réalise

fid int	*pid* int
42	17
38	18
771	70
14	21
499	42
302	60
⋮	

Comme on peut le voir, la table de jonction *Réalise* coïncide exactement avec le losange *réalise* avant. De façon pratique, si une association du diagramme EA possède des attributs, ces derniers peuvent être stockés dans la table de jonction. Cela nous permet de représenter l'association *joue* par la table de jonction :

$$Joue(\text{fid} : int,\ \text{pid} : int,\ \text{role} : text)$$

La table *Joue* peut alors être peuplée de valeurs telles que :

pid int	*fid* int	*role* text
⋮		
3093	499	James Bond
65	499	Vesper Lynd
10765	499	Le Chiffre
6808	499	Felix Leiter
15954	499	René Mathis
19536	499	Solange Dimitrios
4884	499	Alex Dimitrios

Dans une telle table, obtenir tous les rôles du film *Casino Royale* consiste donc simplement à récupérer toutes les lignes ayant 499 comme valeur pour la colonne *fid* (499 étant la valeur de l'identifiant de ce film).

Un dernier point à aborder est celui des attributs à valeurs multiples. Ce dernier est semblable au cas des associations $1 : N$. Pour le cas des pays et des genres, il suffit simplement de créer deux nouvelles relations :

$$Genre(\text{fid} : int,\ \text{genre} : text) \qquad Pays(\text{fid} : int,\ \text{pays} : text)$$

Il devient ainsi possible d'associer à chaque film un nombre arbitraire de genres et de pays. Nous résumons ainsi le schéma global de notre base de données :

$$Personne(pid : int,\ nom : text,\ prénom : text)$$
$$Film(fid : int,\ titre : text,\ année : int,\ durée : int,\ titre\_orig : text)$$
$$Remporte(fid : int,\ année : int,\ pres\_id : int)$$
$$Réalise(fid : int,\ pid : int)$$
$$Joue(fid : int,\ pid : int,\ role : text)$$
$$Genre(fid : int,\ genre : text)$$
$$Pays(fid : int,\ pays : text)$$

La colonne *titre_orig* dans la table *Film* est `NULL` lorsque le titre original est absent.

Pour les quatre dernières tables, l'ensemble des attributs joue à chaque fois le rôle de clé. Par exemple, pour la relation *Réalise*, c'est le couple (fid, pid) qui est unique pour chaque ligne (une personne ne pouvant pas réaliser deux fois le même film). Nous avons indiqué ce fait en soulignant la totalité des attributs. Cette situation est fréquente pour les tables de jonction, dont le seul but est de stocker des associations entre entités.