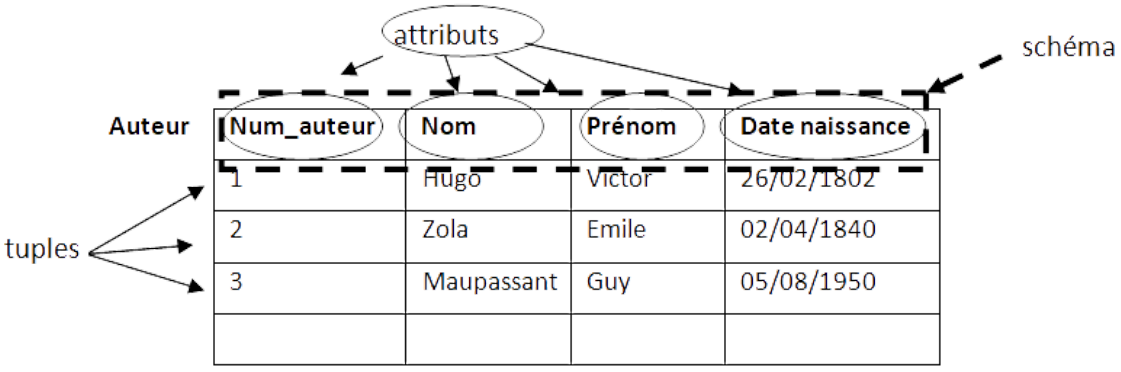
Conception logique : le modèle relationnel

Le modèle relationnel proposé par Edgard Franck Codd en 1970 est la traduction d'une modélisation conceptuelle en une conception logique. C'est le modèle de conception logique le plus utilisé mais ce n'est pas le seul, d'autres existent.

Cours

Dans le modèle relationnel, toutes les **entités-types** et les **associations** de la modélisation conceptuelle sont transformées en tableaux à deux dimensions, appelés **relations**.

Reprenons l'exemple précédent de l'entité-type `Auteur`. Elle se traduit dans le modèle relation par une relation `Auteur` dont les données peuvent être représentées sous la forme tableau, chaque ligne est appelée un **tuple** et chaque colonne un **attribut**.



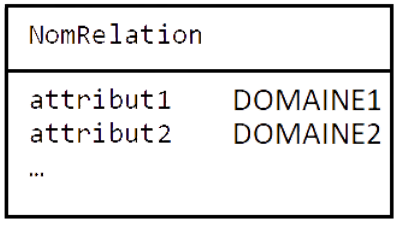
Num_auteur	Nom	Prénom	Date naissance
1	Hugo	Victor	26/02/1802
2	Zola	Emile	02/04/1840
3	Maupassant	Guy	05/08/1950

Cours

L'ensemble des attributs d'une relation forme le **schéma** de la relation.

Le schéma d'une relation se représente en précisant les attributs et leur domaine :

- sous forme textuelle : `NomRelation(attribut1 : DOMAINE1, attribut2 : DOMAINE2….)`.
- sous forme graphique :



Le schéma de la relation décrit une relation « vide » dans laquelle des données peuvent ensuite être stockées (sous forme de tuples).

Par exemple ici : `Auteur(Num_auteur : INT, Nom : TEXT, Prenom : TEXT, Date_naissance : DATE)`,

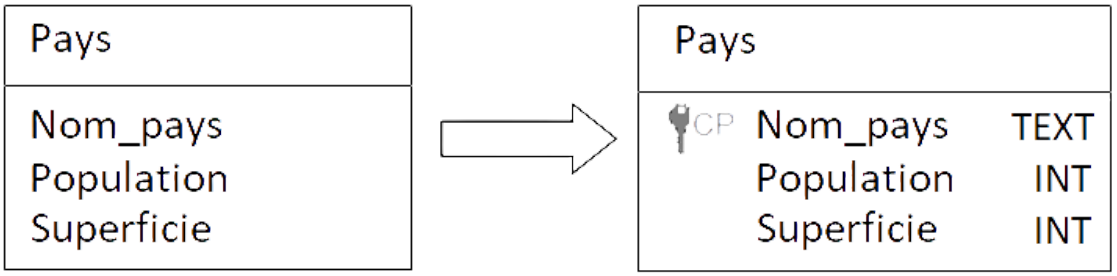
ou encore de façon plus succincte en omettant le domaine des attributs : `Auteur(Num_auteur, Nom, Prenom, Date_naissance)`.

Entité-type représentée par une relation

On transforme une entité-type en relation avec les mêmes attributs. L'identifiant devient la clé primaire.

Cours

Une **clé primaire** est un attribut (ou plusieurs) dont les valeurs permettent de distinguer les tuples les uns des autres de façon unique.



La clé primaire est mise en évidence souvent en **étant soulignée**, ou parfois avec le dessin d'une **clé**, ou à l'aide du symbole **CP** ou encore **PK** (pour *primary key*).

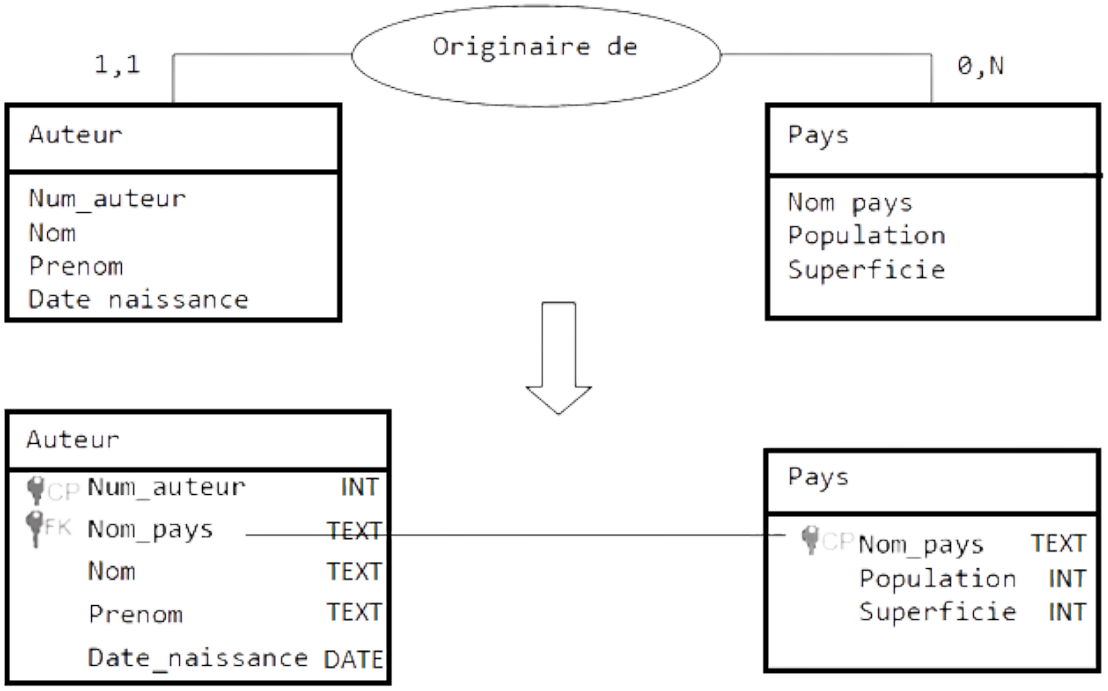
Par exemple ici : `Pays(Nom_pays : TEXT, Population : INT, Superficie: INT)`.

Association représentée par une relation

Association possédant au moins une cardinalité 0,1 ou 1,1

Dans ce cas, chaque tuple de la relation possédant une cardinalité `0,1` ou `1,1` est associé à au plus un tuple de la seconde relation concernée par l'association. Il suffit donc de ajouter un attribut dans la première relation (celle possédant une cardinalité `0,1` ou `1,1`) pour identifier le tuple de la seconde relation. Cet attribut doit permettre d'identifier de façon unique le tuple de la seconde relation, il faut choisir une clé primaire de la seconde relation. Ce nouvel attribut devient une **clé étrangère** de la première relation.

Dans notre exemple, un attribut `Nom_pays` peut-être ajouté dans la relation `Auteur`, cet attribut correspond à la clé primaire de la relation `Pays`. `Nom_pays` est la clé primaire de la relation `Pays` et une clé étrangère de la relation `Auteur`.



A noter: Dans cet exemple, les attributs `Nom_pays` ont le même nom dans les deux relation `Auteur` et `Pays`, mais ils peuvent avoir des noms différents.

Cours

Une **clé étrangère** est un attribut qui est la clé primaire d'une autre relation.

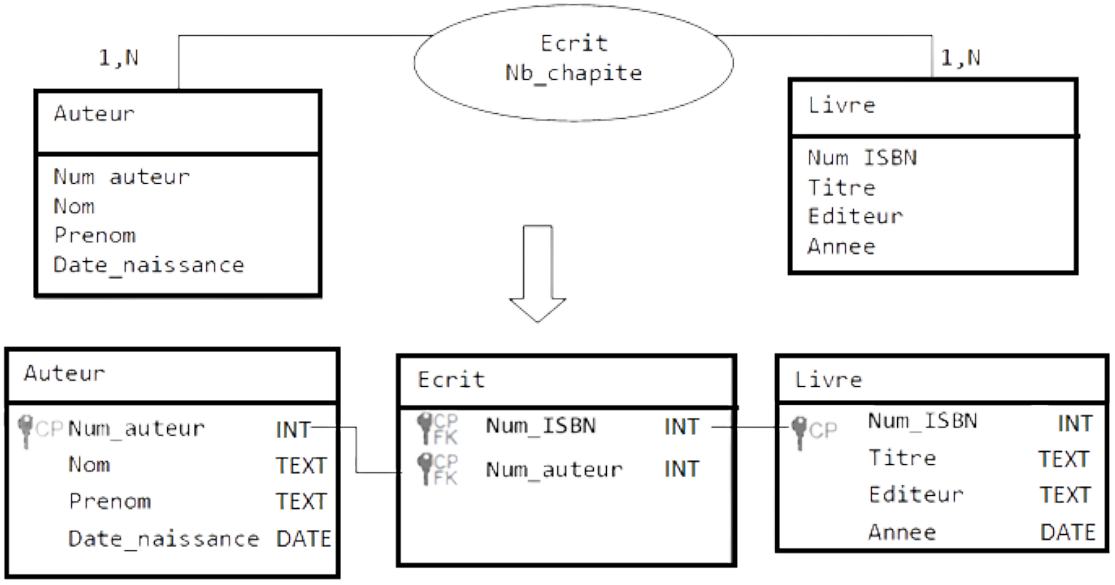
La clé étrangère est mise en évidence souvent avec le symbole #, ou en **étant soulignée en pointillé**, et parfois avec le symbole **FK** (pour *foreign key*).

Par exemple ici : `Auteur(Num_auteur : INT, #Nom_pays : TEXT,  Nom : TEXT, Prenom : TEXT, Date_naissance : DATE)`.

Association sans cardinalité 0,1 ou 1,1

Dans ce cas, il faut transformer l'association en une nouvelle relation avec deux attributs correspondant respectivement aux clés primaires de chacune des entités.

Dans notre exemple, une nouvelle relation `Ecrit` est ajoutée possedant deux clés etrangères `Num_ISBN` et `Num_auteur` correspondant aux deux clés primaires des relations `Livre` et `Auteur`. Le couple de ces deux clés étrangères forment la clé primaire de la nouvelle relation `Ecrit`.



Cours

Une nouvelle relation est ajoutée, possedant **deux clés étrangères** qui forment ensembles la clé primaire de cette relation.

En notation textuelle, les deux clés étrangères de la nouvelle relation sont indiquées par le caractère # (ou soulignées en pointillé) et elles sont toutes les deux soulignées par un trait continu pour indiquer qu'elles forment ensembles la clé primaire.

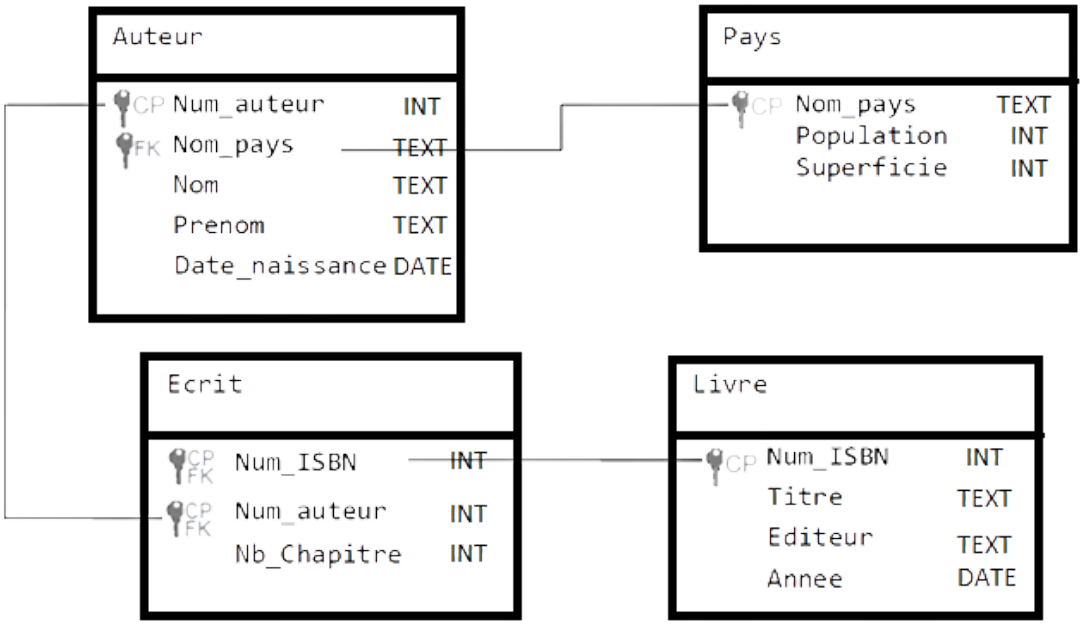
Dans notre exemple : `Ecrit(#Num_ISBN : INT, #Num_auteur : INT)`.

Schéma relationnel d'une base de données

Cours

Le **schéma relationnel d'une base de données** est composé de l'ensemble des relations qui la composent avec leur schéma respectif, ainsi que les contraintes d'intégrité associées à ces relations.

Ce qui donne dans notre exemple sous forme graphique :



et en notation textuelle : `Auteur(Num_auteur : INT, #Nom_pays : TEXT, Nom : TEXT, Prenom : TEXT, Date_naissance : DATE)`

`Pays(Nom_pays : TEXT, Population : INT, Superficie : INT)`

`Livre(Num_ISBN : INT, Titre TEXT, Editeur TEXT, Annee DATE)`

`Ecrit(#Num_ISBN : INT, #Num_auteur : INT)`

A noter :

- deux relations ne peuvent pas avoir le même nom dans une base de données (car une relation est identifiée par son nom), et
- deux attributs ne peuvent pas avoir le même nom dans une relation, mais
- deux attributs appartenant à des relations différentes peuvent avoir le même nom sans être liés. Par exemple on aurait pu utiliser Nom au lieu de Nom_pays dans la relation Pays.

Les contraintes d'intégrité

Cours

Il existe un certain nombre de **règles à respecter** pour respecter l'intégrité d'une base de données. Ces règles visent à préserver la cohérence des données et garantir une stabilité de notre base dans le temps.

Contraintes d’entité ou unicité des clés

Cours

Il ne peut y avoir de doublons dans une relation. Toute relation doit posséder un identifiant unique appelé clé primaire.

Le problème typique est l'utilisation de l'attribut Nom dans notre entité Auteur. Cet attribut ne peut définir de manière unique un auteur car plusieurs auteurs peuvent avoir le même nom, ce ne peut pas être une clé primaire. On utilise un attribut Num_auteur qui doit être différent pour chaque tuple afin que chaque auteur soit identifié par un numéro unique.

Contraintes référentielles

Les clés primaires distinguent de manière unique chaque tuple mais peuvent également servir de références dans d'autres relations (clé étrangères). Il faut veiller à ce que les références soient effectives.

Cours

Il n'est pas possible de définir une entité qui fait référence par une clé étrangère à une entité qui n'existe pas.

Reprenons la relation `Ecrit(Num_ISBN, Num_auteur)` où `No_ISBN` et `Num_auteur` sont des clés étrangères. Tous les livres doivent avoir un auteur connu. Pour ajouter un nouveau livre, il faut que l'auteur soit existant et respectivement il doit être impossible de supprimer un auteur si un de ses livres est encore présent dans `Livre`.

Contraintes de domaine et contraintes utilisateurs

Le **domaine** d'un attribut permet de préciser le type de données stockées .

Chaque SGBD offre ses propres types de données qui regroupent en général les types de données habituels des langages de programmation (integer, booléens, float, string, etc.), et des types supplémentaires par exemple les dates ou des données binaires comme des images ou vidéos (le type BLOB pour *binary large object*).

Les noms des types de données peuvent varier d'un SGBD à l'autre, par exemple on trouve `CHAR`, `VARCHAR(x)` ou `TEXT` pour les chaîne de caractère.

Cours

En plus du type de données, le **domaine peut contenir des contraintes supplémentaires ou contraites utilisateurs** sur les données afin d'éviter les erreurs de saisies.

Par exemple, le domaine :

- une donnée doit appartenir à une liste (liste des communes de France par exemple) ;
- une donnée numérique doit être bornée (âge d'un personne) ;
- possède un nombre de caractères défini à l'avance (numéro de sécurité sociale, code postal) ;
- etc.