Huitième partie VIII

Production du schéma de la base de données

Plan du cours

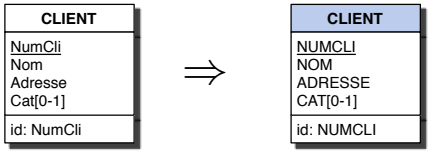
- Partie I : Introduction aux bases de données relationnelles
  - Cours 1 : Concepts des bases de données relationnelles
  - Cours 2 : L'algèbre relationnelle
- Partie II : Utilisation des bases de données relationnelles
  - Cours 3 : Le langage SQL DML (1)
  - Cours 4 : Le langage SQL DML (2)
  - Cours 5 : Le langage SQL DDL
- Partie III : Developpement des bases de données relationnelles
  - Cours 6 : Le modèle entité-association
  - Cours 7 : Élaboration d'un schéma conceptuel
  - Cours 8 : Production du schéma de la base de données

Qu'allons nous aborder dans ce cours ?

1. Le principe de la représentation d'un schéma conceptuel dans le langage de définition de donnée d'un SGBD, i.e., SQL DDL
2. Patriquement, on s'intéresse à la représentation :
   - Des types d'entités
   - Des attributs
   - Des associations
   - Des identifiants

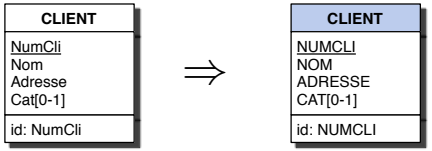
Représentation des types d'entités

- Chaque type d'entités est représenté par une table à la quelle on donne le nom de ce type d'entités
- Chaque entité de ce type sera représentée par une ligne dans la table
- Exemple :



Représentation des attributs

- Chaque attribut d'un type d'entités est représenté par un colonne de la table qui représente le type d'entité
- Le type et la longueur des valeurs de la colonne sont définis en fonction du domaine de valeurs de l'attribut
- La colonne est obligatoire ou facultative selon l'attribut qu'elle représente est lui-même obligatoire ou facultatif
- Le nom d'une colonne doit se conformer à la syntaxe imposée par le langage SQL
- Exemple :



Représentation des types d'associations

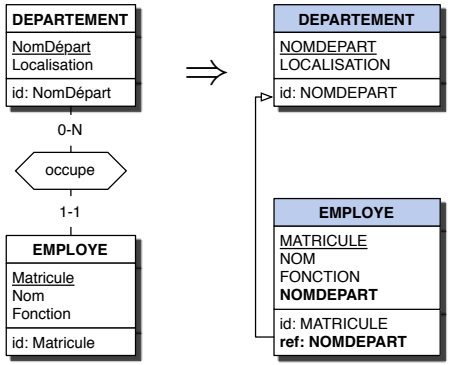
- La représentation des types d'assocations est un peu plus complexe
- Nous pouvons distinguer trois classes fonctionnelles de types d'association :
  1. *un-à-plusieurs*
  2. *un-à-un*
  3. *plusieurs-à-plusieurs*

Représentation des types d'associations

- Soit
  - R un type d'association *un-à-plusieurs* entre A et B (plusieurs entités de B pour chaque entité A, une seule entité A pour chaque entité B)
  - L'attribut IA est l'identifiant primaire de A
  - A est représenté par la table TA et B par la table TB
- On représente alors R par :
  - Une colonne RA de même type que IA que l'on ajoute à la table TB. Cette colonne RA est déclaré clé étrangère de B vers A.
- Remarque : Si R est obligatoire pour B, la colonne RA de la table B sera déclaré obligatoire. Si en revanche R est facultatif, alors RA sera déclarée facultative

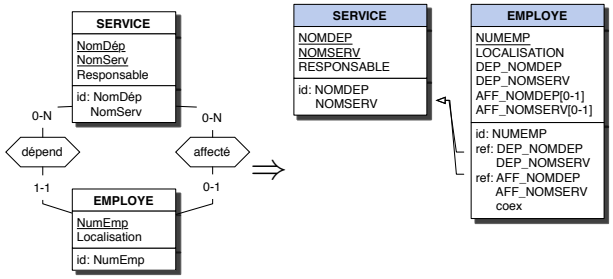
Représentation des types d'associations

- Exemple :



Représentation des types d'associations

- Lorsque l'identifiant de A est constitué de plusieurs attributs $IA1$, $IA2$, ... on définit autant de nouvelles colonnes $RA1$, $RA2$, ... dans la table TB qui forment la clé étrangère
- Exemple :



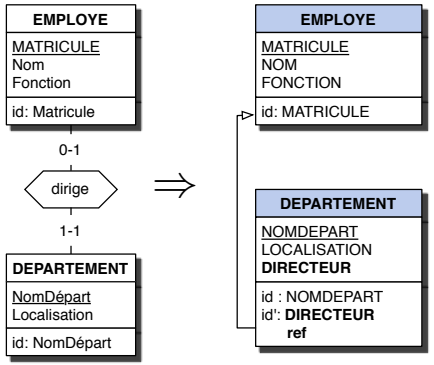
- Remarque : coex indique tous les composant de la clé étrnagère doivent être simultanément null ou non null. Il s'agit d'une contrainte de coexistence.

Représentation des types d'associations

- Soit
  - R un type d'association *un-à-un* entre A et B (une seule entité B pour chaque entité A, une seule entité A pour chaque entité B)
  - A est représenté par la table TA et B par la table TB
  - L'indentifiant primaire de A est IA et/ou celui de B IB
- On représente alors R par :
  - l'ajout à la table d'un type d'entités A ou B d'une colonne de même type que l'identifiant de l'autre table, et on déclare cette colonne clé étrangère vers l'autre table. En outre, cette colonne est déclarée identifiante.
- Remarque : Si l'identifiant de la table référencée est constitué de plusieurs attributs, la clé étrangère sera constituée d'autant d'attributs

Représentation des types d'associations

- Exemple :



Représentation des types d'associations

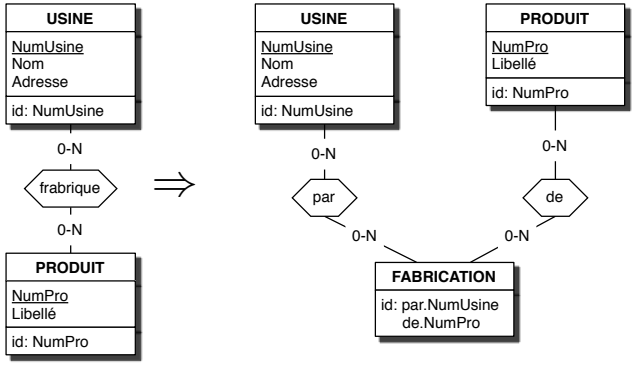
- Plus précisément :
  1. Si R est obligatoire pour A et facultatif pour B, on ajoutera à TA une nouvelle colonne obligatoire RB, de même type que IB et qu'on déclare clé étrangère vers TB
  2. Si R est facultatif pour A et B, une clé étrangère est ajoutée indiffférent à TA et TB. La nouvelle colonne sera déclarée facultative.
  3. Dans tous les cas, la nouvelle colonne (ou les nouvelles colonnes) constitue(nt) en outre un identifiant suplémentaire pour sa table.
  4. Le cas où R est obligatoire oiur A et B n'est pas traité ici.

Représentation des types d'associations

- Soit R un type d'association *plusieurs-à-plusieurs* entre A et B (plusieurs entités B pour chaque entité A, plusieurs entités A pour chaque entité B)
- On représente alors R en :
  1. transformant R en un type d'entité R' et deux types d'association *un-à-plusieurs*
  2. appliquant la méthode de représentation des associations de type *un-à-plusieurs* précédemment décrite

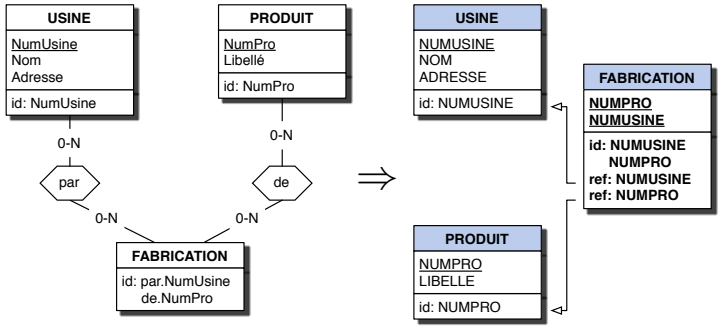
Représentation des types d'associations

- Exemple : Etape 1 (Décomposition en association *un-à-plusieurs*



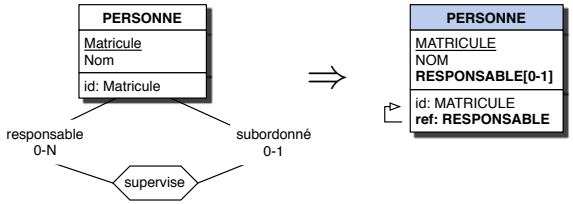
Représentation des types d'associations

- Exemple : Etape 2 (Représentation des associations *un-à-plusieurs*



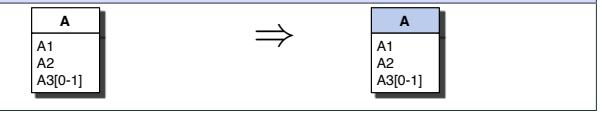
Représentation des types d'associations

- Les régles de représentation précédemment énoncées s'appliquent aux types d'association cyclique
- Exemple : Type d'associations cyclique de type *un-à-plusieurs*



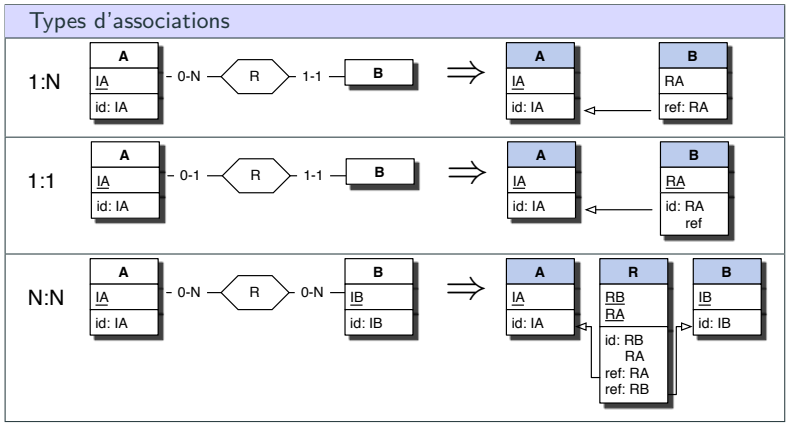
Synthèse des règles de traduction

Types d'entités et attributs



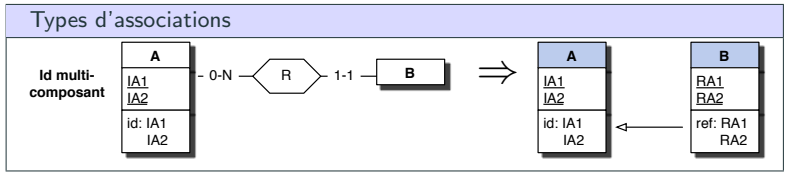
Synthèse des règles de traduction

Types d'associations

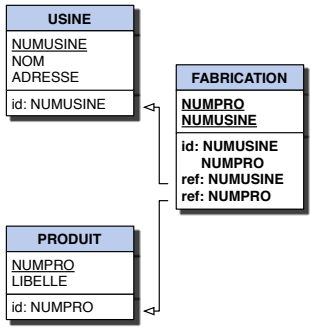


Synthèse des règles de traduction

Types d'associations



Traduction des structures en SQL



SQL

```
create schema prodution;

create table USINE ( NUMUSINE char(10) not null,
                     NOM char(30) not null,
                     ADRESSE char(40) not null,
                primary key (NUMUSINE) ) ;

create table PRODUIT ( NUMPRO char(10) not null,
                       LIBELLE char(40),
                primary key (NUMPRO) ) ;

create table FABRICATION ( NUMPRO char(10) not null,
                           NUMUSINE char(10) not null,
                primary key (NUMPRO, NUMUSINE),
                foreign key (NUMPRO) reference PRODUIT
                      on delete cascade on update cascade,
                foreign key (NUMUSINE) reference USINE
                      on delete cascade on update cascade) ;
```