Chapitre 5

LE MODELE ENTITE - ASSOCIATION

Introduction

Conception d'une base de données

- Domaine d'application complexe :
  - description abstraite des concepts indépendamment de leur implémentation sous forme de tables
- Analyse conceptuelle :
  - définition d'un schéma conceptuel
  - => modèle entité-association
- Production de la base de données :
  - Traduction du schéma conceptuel en tables
  - => schéma SQL
- Ateliers de génie logiciel (AGL) spécialisés
  - Ex : MERISE, UML

1. Description du modèle entités-associations [Hainaut]

1.1. Types d'entités

- Le domaine d'application est constitué d'entités concrètes ou abstraites
- Chaque entité appartient à une classe ou type d'entité
- Exemple : domaine de l'assurance automobile
  - Objet concret animé : CLIENT
  - Objet concret inanimé : VEHICULE
  - Objet abstrait : CONTRAT
  - Evenement : ACCIDENT
  - Représentation graphique
- Chaque type d'entité représente une population, variable dans le temps

1.2. Attributs

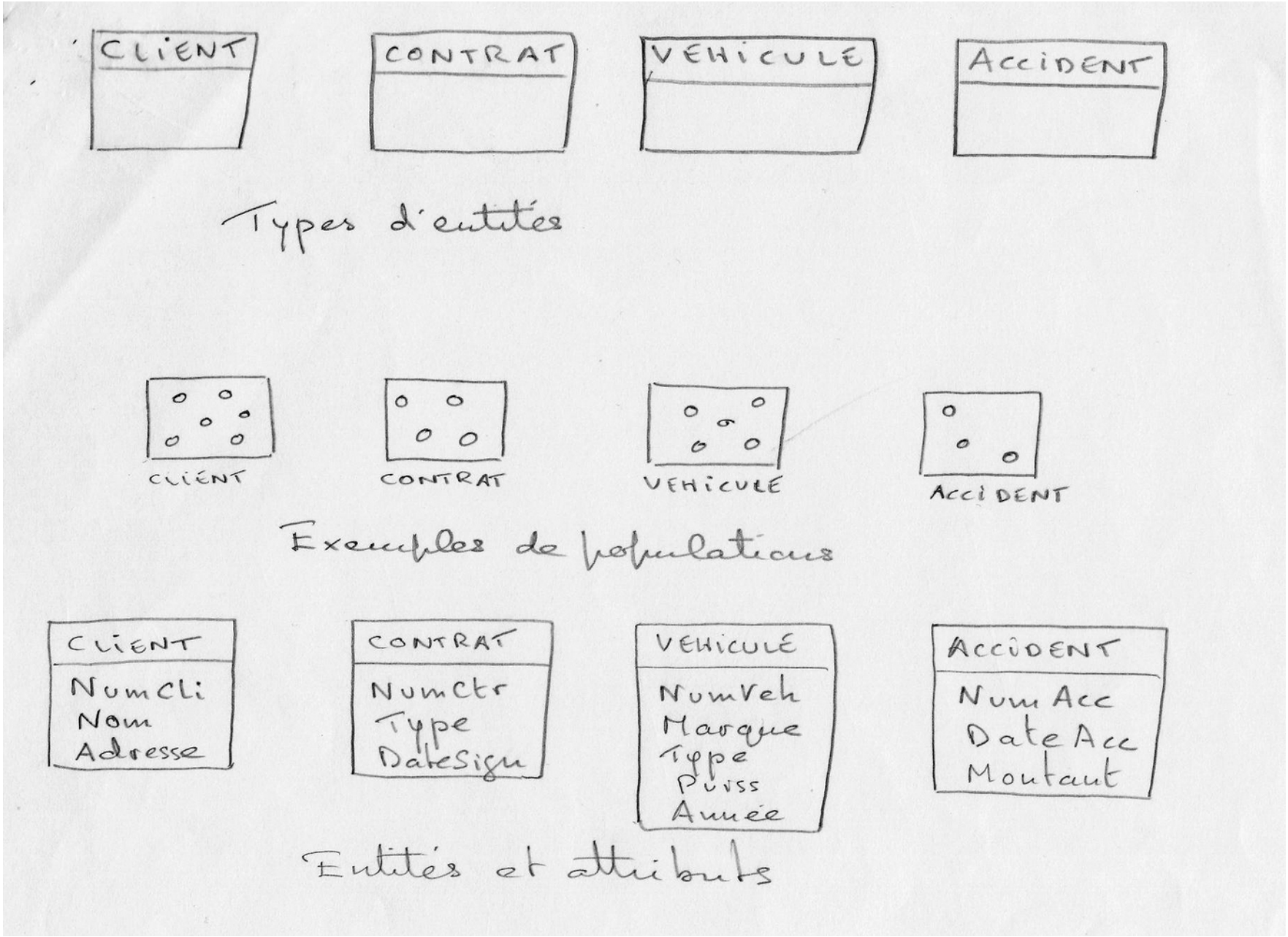
- Chaque type d'entité est caractérisé par des propriétés ou attributs
- Il faut spécifier le type de chaque attribut et sa longueur:
  - Numérique, caractère, date, etc.
  - Exemple
- Lorsque l'on décrit une entité, la valeur d'un attribut :
  - soit peut être inconnue :

    => attribut facultatif de cardinalité [0-1]
  - soit doit être connue :

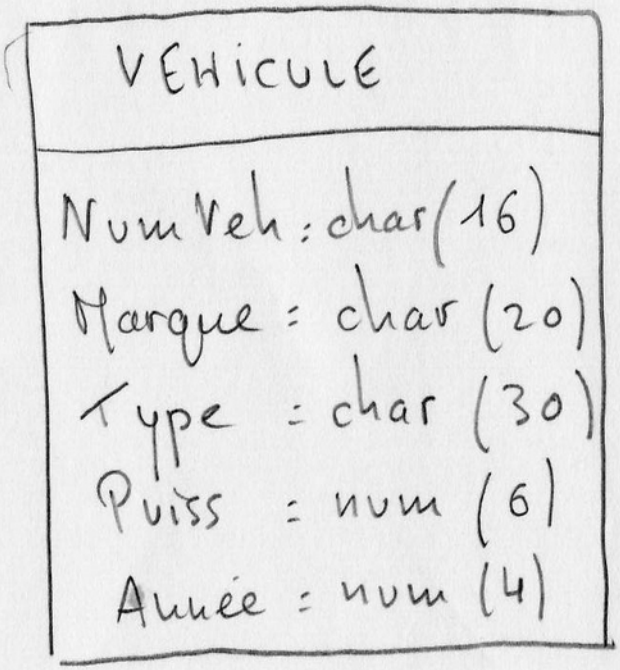
    => attribut obligatoire de cardinalité [1-1]

    (cas par défaut)

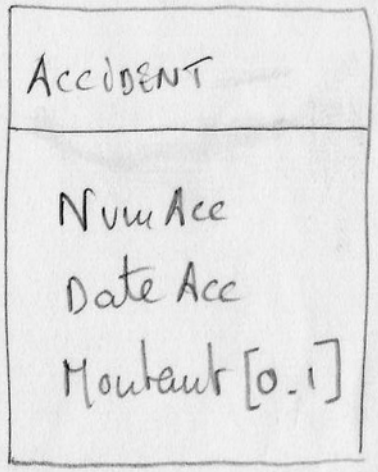
Exemple d'entités, populations et attributs



Exemple de types des attributs



Types de valeurs des attributs



Attributs obligatoires et facultatifs

1.3. Types d'associations

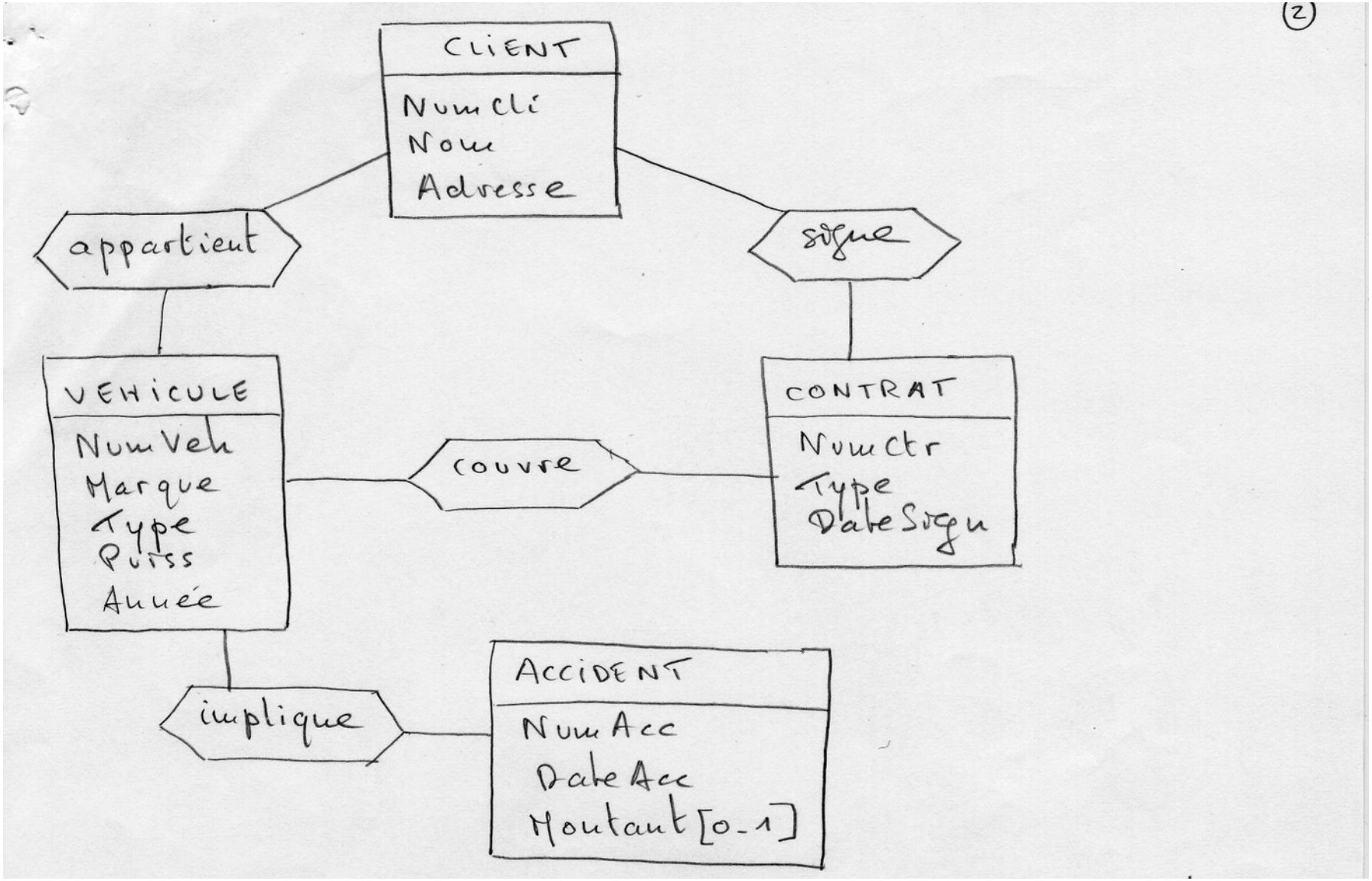
- Caractérisent les liens entre les types d'entités

- Exemple :
  - Association *signe* entre CLIENT et CONTRAT
  - Association *appartient* entre CLIENT et VEHICULE
  - Association *couvre* entre CONTRAT et VEHICULE
  - Association *implique* entre ACCIDENT et VEHICULE

- Lorsqu'un type d'entité intervient dans un type d'association, on dit qu'il y joue un rôle

- Un même type d'entité peut jouer deux rôles différents dans un même type d'association

Exemple d'entités et d'associations



1.3.1. Propriétés d'un type d'association

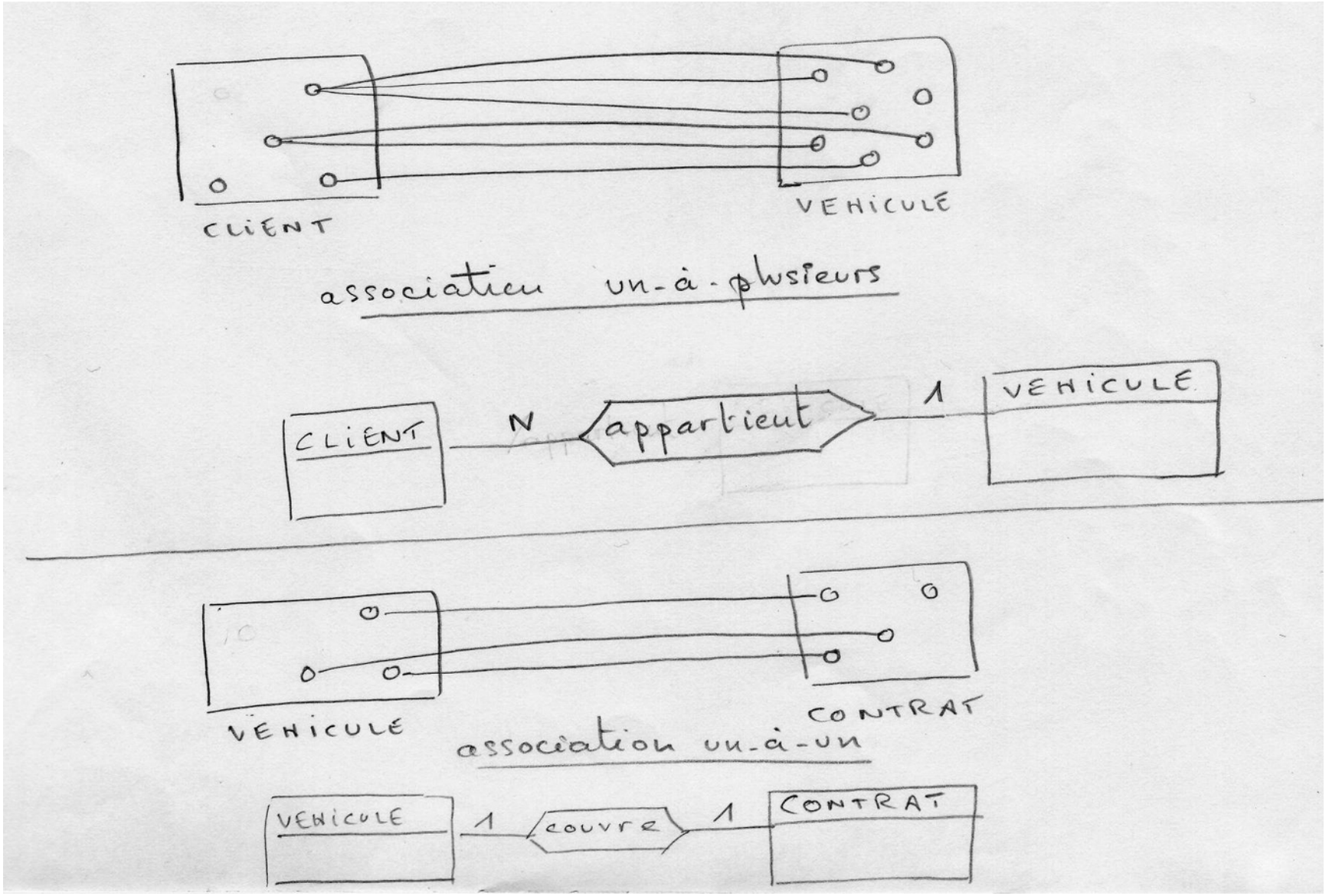
- Soit un type d'association R entre des types d'entités A et B
- Objectif : déterminer le nombre minimum et maximum d'associations R auxquelles tout entité A et B participe
- Ces propriétés de cardinalité sont classées en deux catégories :
  - Classe fonctionnelle
  - Caractère obligatoire/facultatif

a. Classe fonctionnelle d'un type d'association

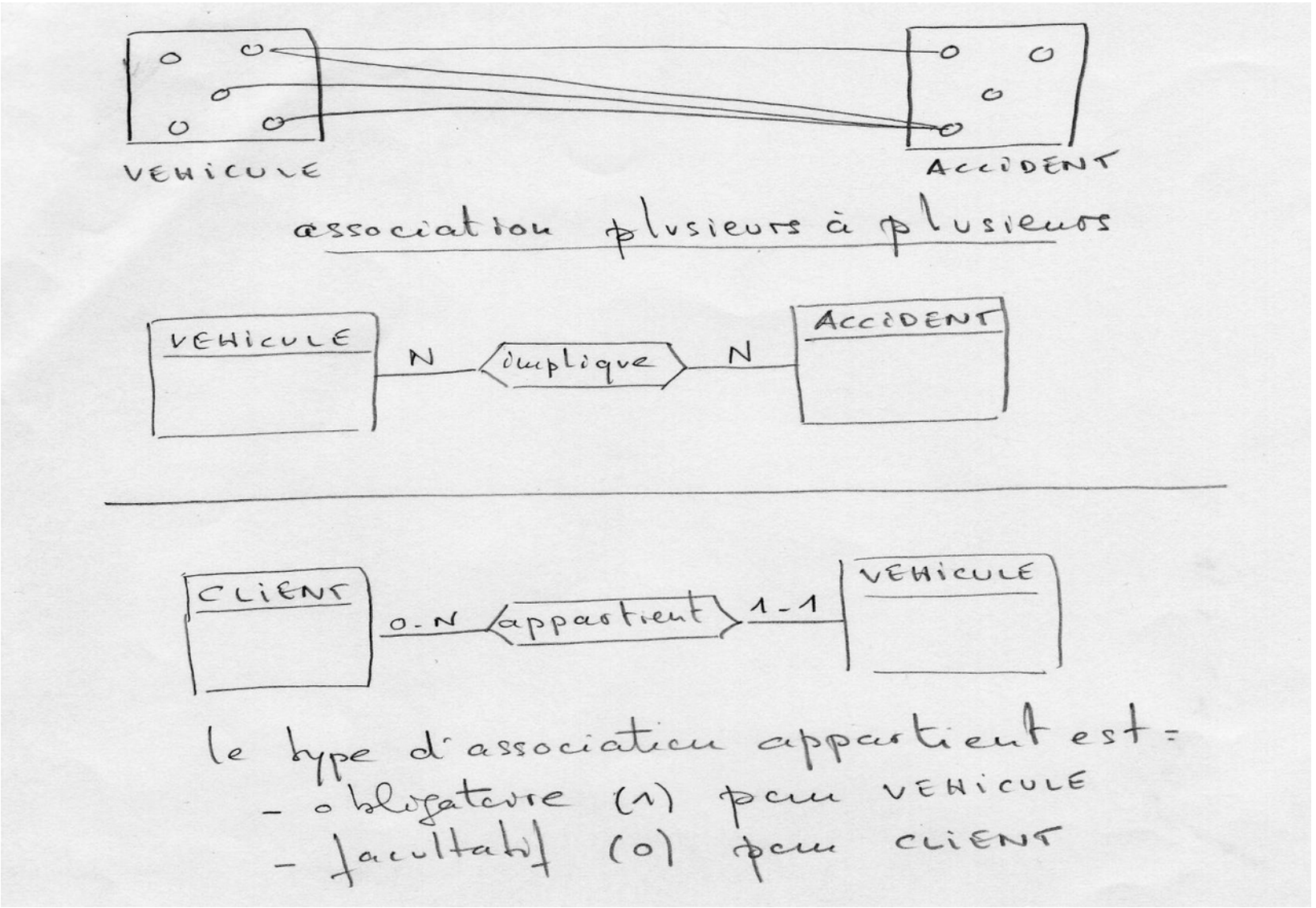
- Cette propriété décrit le nombre maximum d'entités B pour chaque entité A et inversement. On définit trois classes :
  - *un-à-plusieurs*
  - *un-à-un*
  - *plusieurs-à-plusieurs*
- On indique par 1 ou N le nombre maximum d'arcs associés

- *Type d'association un-à-plusieurs*
  - Exemple : entités CLIENT et VEHICULE
  - « un véhicule appartient à un seul client »
  - « un client peut posséder plusieurs véhicules »
  - Représentation graphique
- *Type d'association un-à-un*
  - Exemple : entités VEHICULE et CONTRAT
  - « un contrat couvre un seul véhicule »
  - « un véhicule est couvert par un seul contrat »
  - Représentation graphique
- *Type d'association plusieurs-à-plusieurs*
  - Exemple : entités VEHICULE et ACCIDENT
  - « un véhicule peut être impliqué dans plusieurs accidents »
  - « un accident peut impliquer plusieurs véhicules»
  - Représentation graphique

Exemple de cardinalités des associations



Exemple de cardinalités des associations



b. Type d'association obligatoire ou facultatif

- On indique par O ou 1 le nombre minimum d'arcs issus de toute entité
- Un type d'association peut être obligatoire pour un type d'entité qui y participe
- Exemple : entités CLIENT et VEHICULE
  - « tout véhicule appartient à un client »
  - *appartient* est déclaré obligatoire pour VEHICULE
  - Représentation graphique
  - Toute entité VEHICULE est associée via *appartient* à exactement une (de 1 à 1) entité CLIENT
  - Toute entité CLIENT est associée via *appartient* à un nombre quelconque (de 0 à N) d'entités VEHICULE

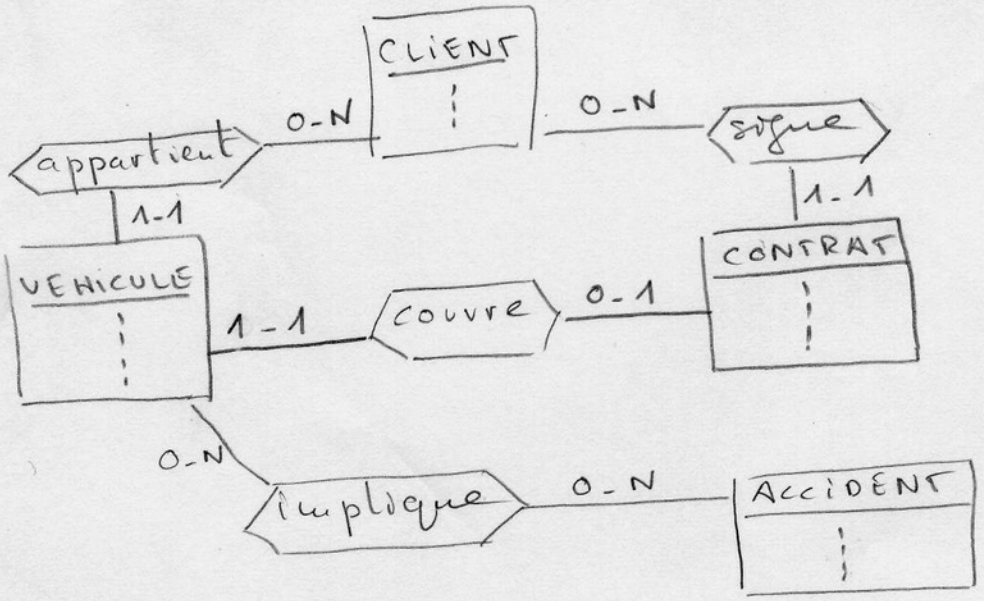
c. Cardinalités d'un type d'association

- Chaque type d'entité jouant un rôle dans un type d'association, y est caractérisée par un couple de valeurs *min-max* appelé cardinalité.

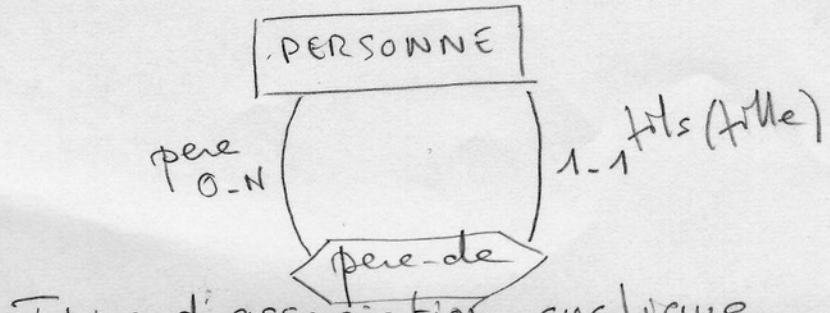
- On distingue trois valeurs de cardinalité :
  - 0-1 (au plus un)
  - 1-1 (exactement un)
  - 0-N (nombre quelconque)
  - Ex : *appartient* est de cardinalité [0-N,1-1] de CLIENT vers VEHICULE
  - Représentation graphique de l'exemple
  - Exemple d'association cyclique au sein de l'entité PERSONNE : *père_de* de cardinalité [0-N,1-1]

    -> Dans cette association, une personne joue le rôle de père et l'autre celui de fils (ou fille)

Exemple de cardinalités des associations

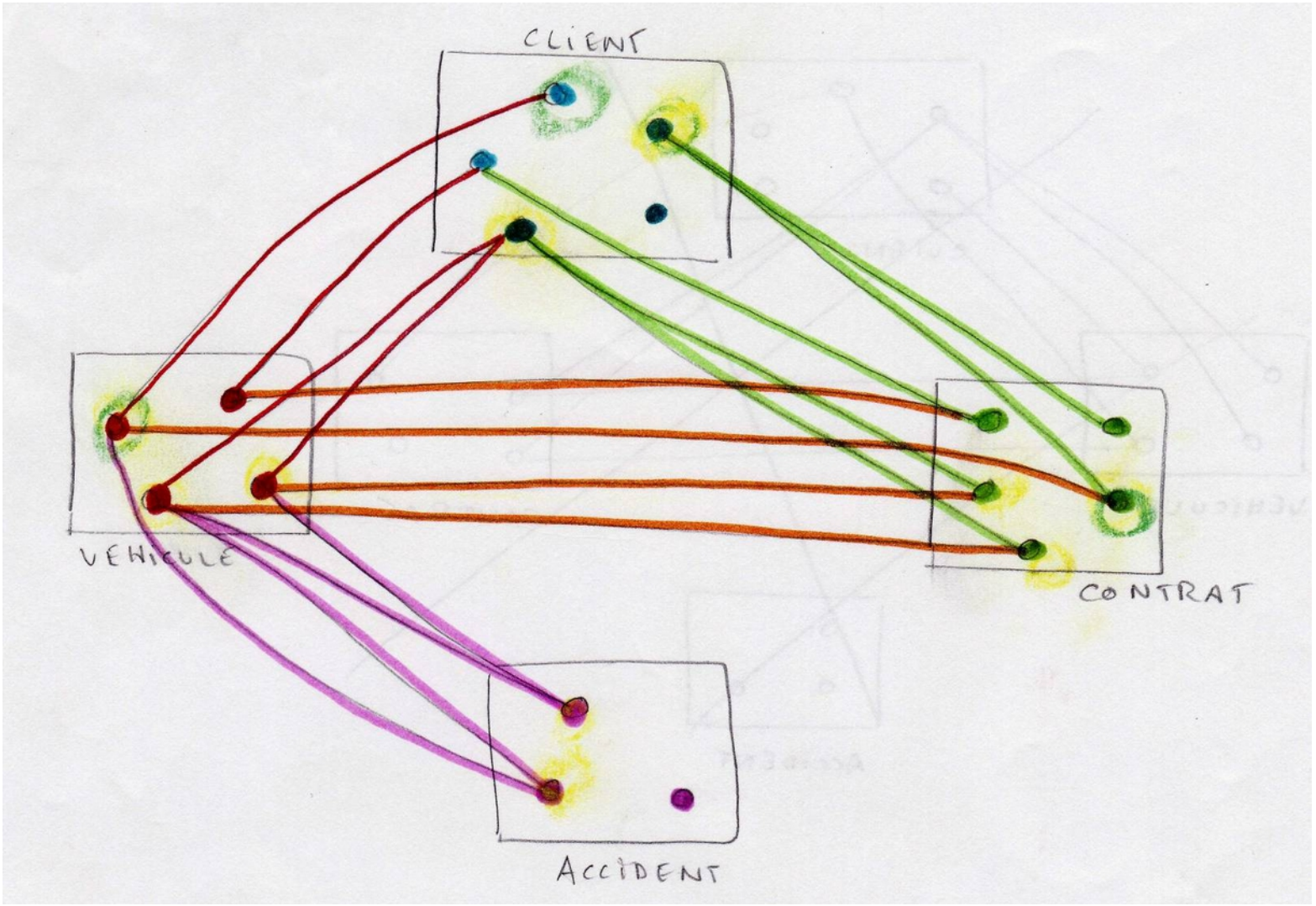


Cardinalités complètes



Type d'association cyclique

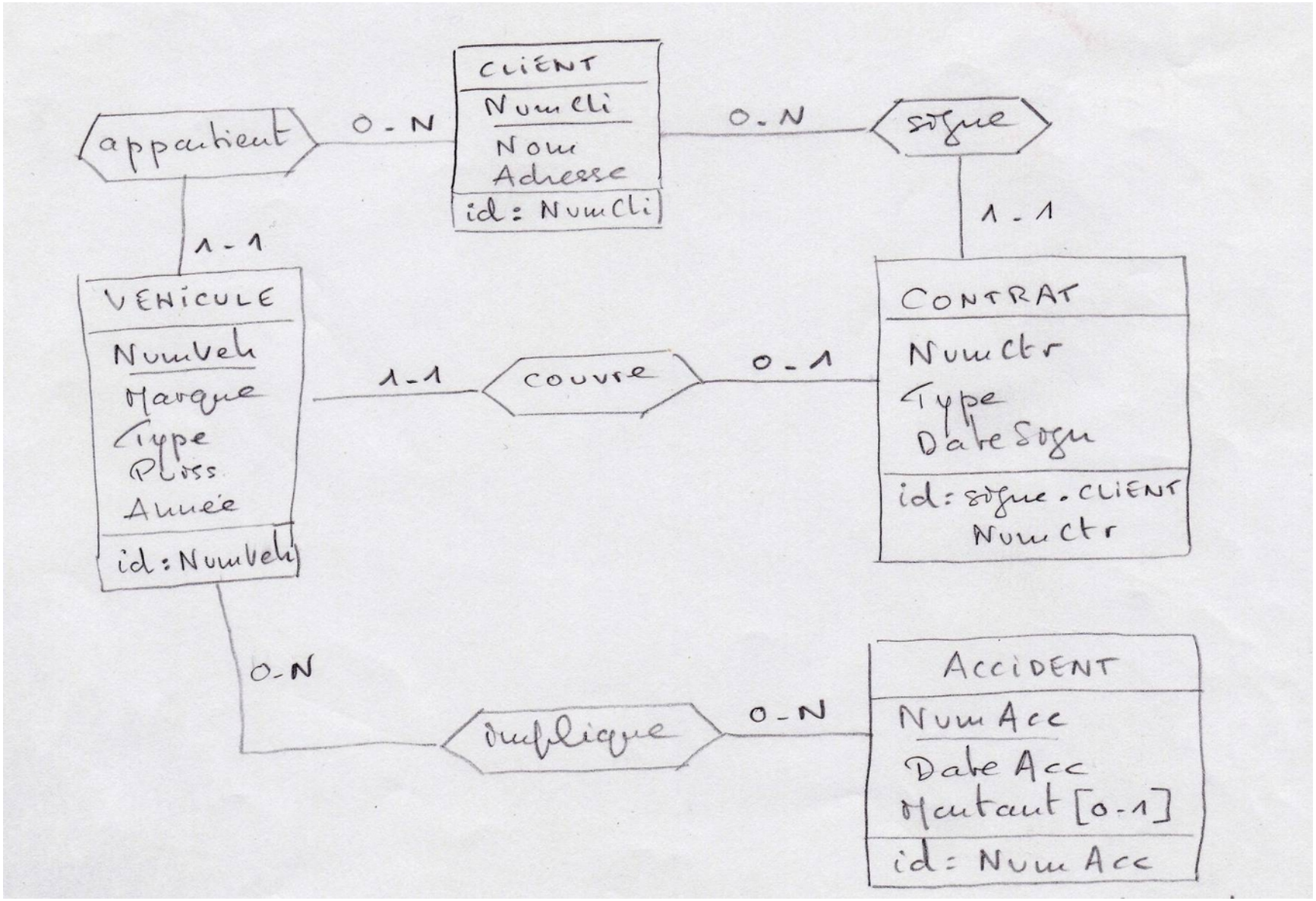
Exemple de représentation graphique des populations



1.4. Les identifiants

- En général un type d'entité est doté d'un identifiant :
  - attribut qui distingue les entités de ce type
  - Ex : NumClient est un identifiant de CLIENT
- Un identifiant peut être formé de plusieurs attributs
- Un type d'entité peut posséder plusieurs identifiants :
  - id: <identifiant primaire>
  - id': <identifiant secondaire>
  - Exemple
- Cas des identifiants hybrides
  - Constitués de types d'entités voisins et d'attributs
  - Exemple : CONTRAT est identifié par (CLIENT,NumCtr)

Illustration de la notion d'identifiant



- Un identifiant doit être minimal, i.e qu'aucun de ses composants ne peut être retiré sans qu'il perde son statut d'identifiant

- Un identifiant primaire doit être chosi tel que ses valeurs soient stables pendant la durée de vie de l'entité
  - Ex : éviter <id_école>=<code postal> + <initiales du nom de l'établissement>
  - => pb si changement de nom ou de code postal

1.5. Les contraintes d'intégrité

Ce sont les propriétés que les objets décrits par un schéma (entités, valeurs d'attributs, associations) doivent respecter afin de respecter le comportement du domaine d'application

- Contraintes d'intégrité statiques

  décrivent les états valides :

  – Exemples :

  - le prix unitaire doit être supérieur à 0
  - toute commande doit avoir au moins un détail

- Contraintes d'intégrité dynamiques

  spécifient des changements d'états valides :

  – Exemples :

  - on ne peut augmenter le prix d'un produit de plus de 5%
  - le changement d'état civil permet de passer de « célibataire » à « marié », mais pas de « célibataire » à « veuf » ou à « divorcé »

1.6. Contenu informationnel d'un schéma

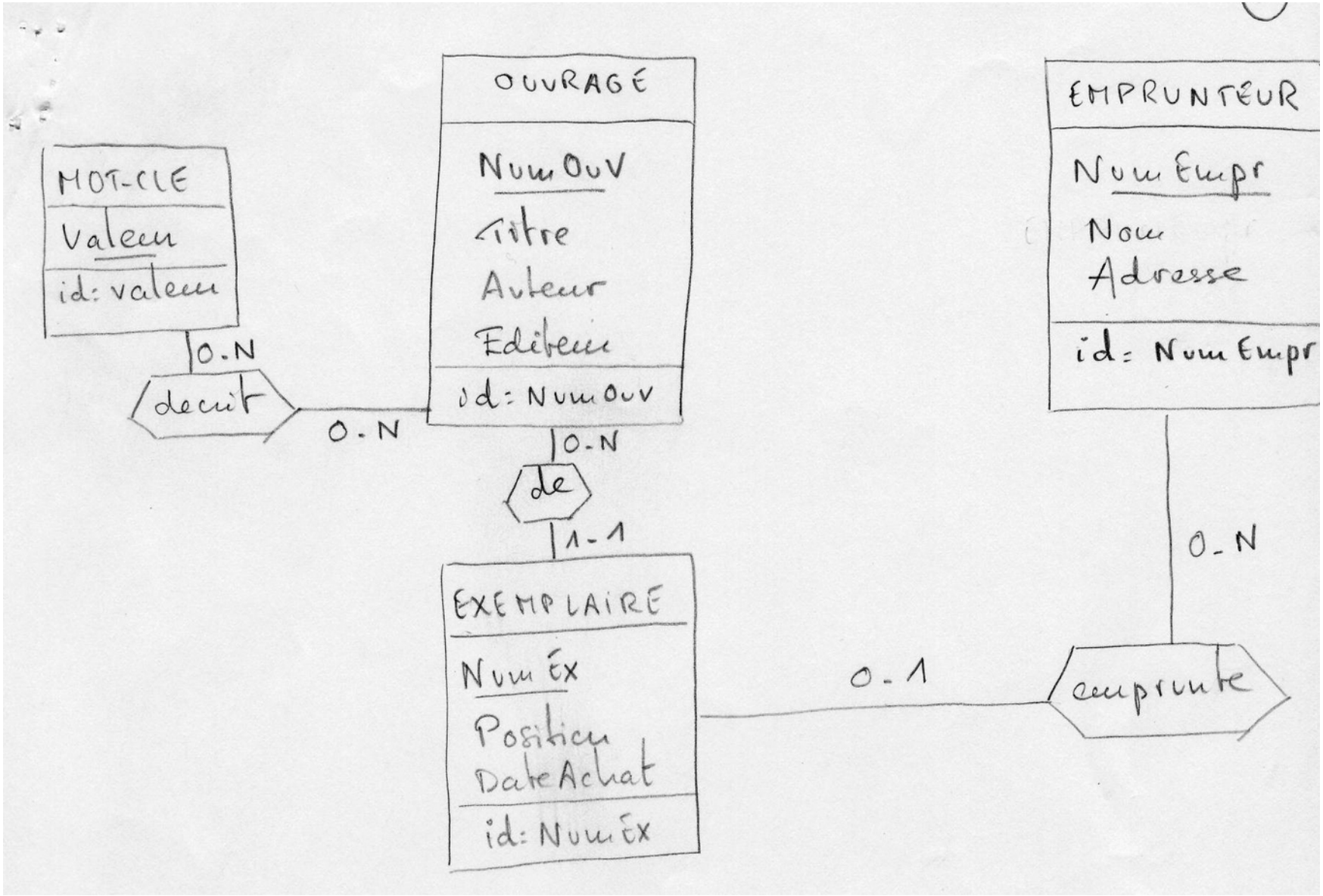
- La représentation graphique des populations permet de vérifier qu'un schéma est capable de répondre à des questions que se posent les utilisateurs

- Exemple :
  - Trouver les signataires des contrats couvrant les véhicules impliqués dans un accident déterminé
  - Trouver les véhicules dont le propriétaire n'est pas le client ayant signé le contrat qui les couvre
  - => voir représentation graphique

2. Exemple : gestion d'une bibliothèque [Hainaut]

- Une bibliothèque contient des ouvrages pouvant être empruntés
- Un ouvrage est caractérisé par un numéro identifiant, un titre, un auteur, un éditeur
- Un ouvrage est décrit par un ensemble de mots-clés qui indiquent les sujets traités
- La bibliothèque dispose d'un ou plusieurs exemplaires de chaque ouvrage
- L'exemplaire d'un ouvrage est identifié par un numéro et caractérisé par sa position dans les rayonnages et sa date d'achat
- Un exemplaire peut être emprunté par un emprunteur
- Un emprunteur est identifié par un numéro, et caractérisé par son nom et son adresse

Exemple : gestion d'une bibliothèque



Exemple : gestion d'une bibliothèque

Ce schéma permet de répondre aux questions :

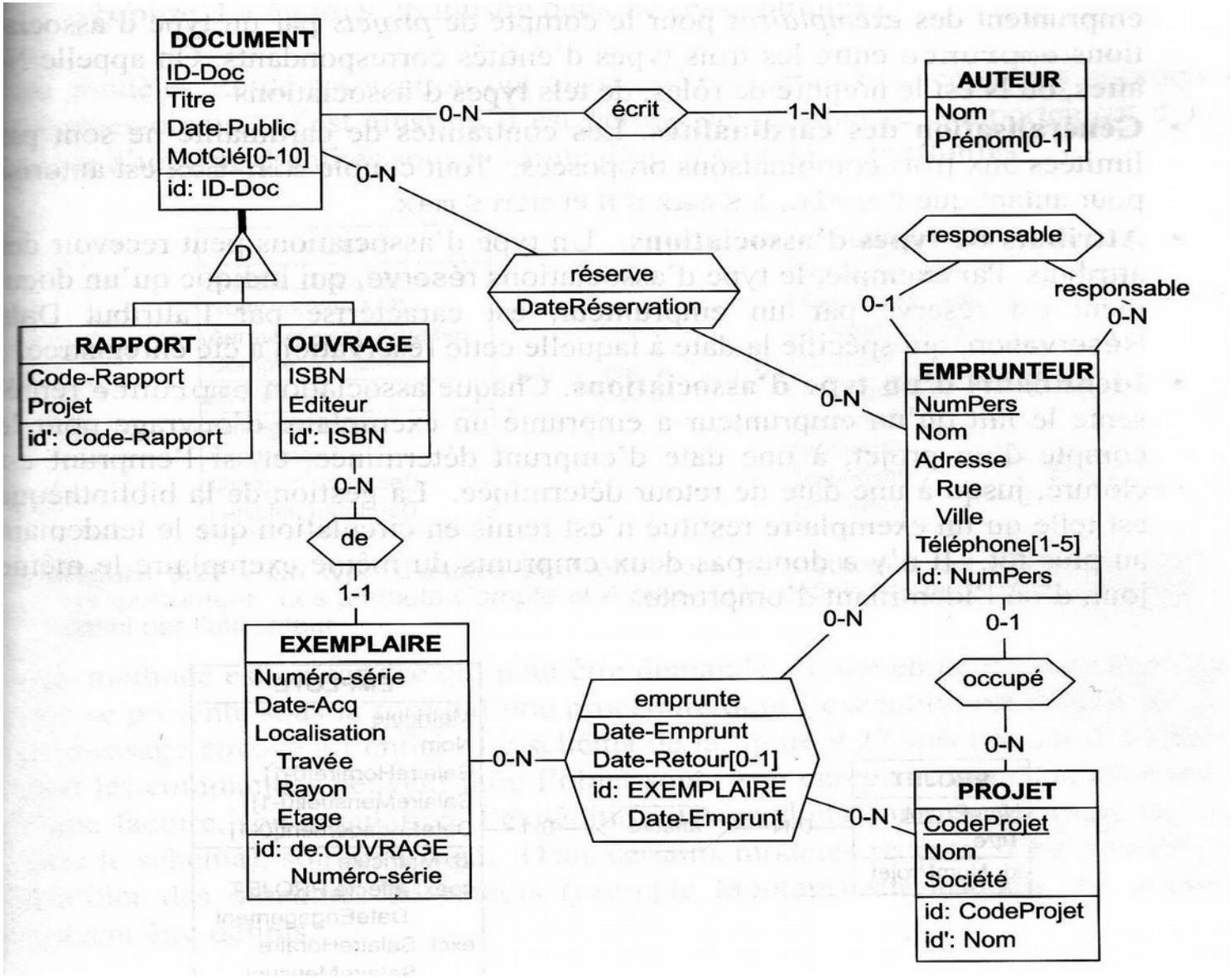
- Quels auteurs lit un emprunteur ?
- Quels sont les emprunteurs intéressés par un certain sujet (valeur du mot-clé) ?
- Quels sont les ouvrages empruntés par plus de deux emprunteurs ?
- Quels sont les ouvrages pour lesquels il reste des exemplaires disponibles (non empruntés) ?

3. Extensions du modèle entité-association

- Relations sur-type/sous-type ou IS-A
- Un type d'entité peut ne pas avoir d'identifiant
- Types d'associations N-aires
- Généralisation des cardinalités
- Attributs de types d'associations
- Identifiants de types d'associations
- Attributs composés
- Attributs multivalués
- Contraintes d'intégrité
- Concepts orientés objets (méthodes)
  - =>modèle UML (Universal Modelling Language)
    basé sur l'approche orientée objet

Exemple étendu de gestion d'une bibliothèque

Exemple de schéma E-A étendu |Hainaut]



4. Production du schéma de la BD

⇒*Traduction du schéma conceptuel dans le langage de définition des données de la BD (ex: SQL)*

4.1. Représentation des types d'entités

- Chaque type d'entités est représenté par une table portant le même nom
- Chaque entité de ce type correspond à une ligne de la table
- Exemple

4.2. Représentation des types d'associations

- R : type d'associations entre A et B
- TA et TB : tables de A et de B
- IA et IB : identifiants primaires de A et de B

4.2.1. Types d'associations un-à-plusieurs

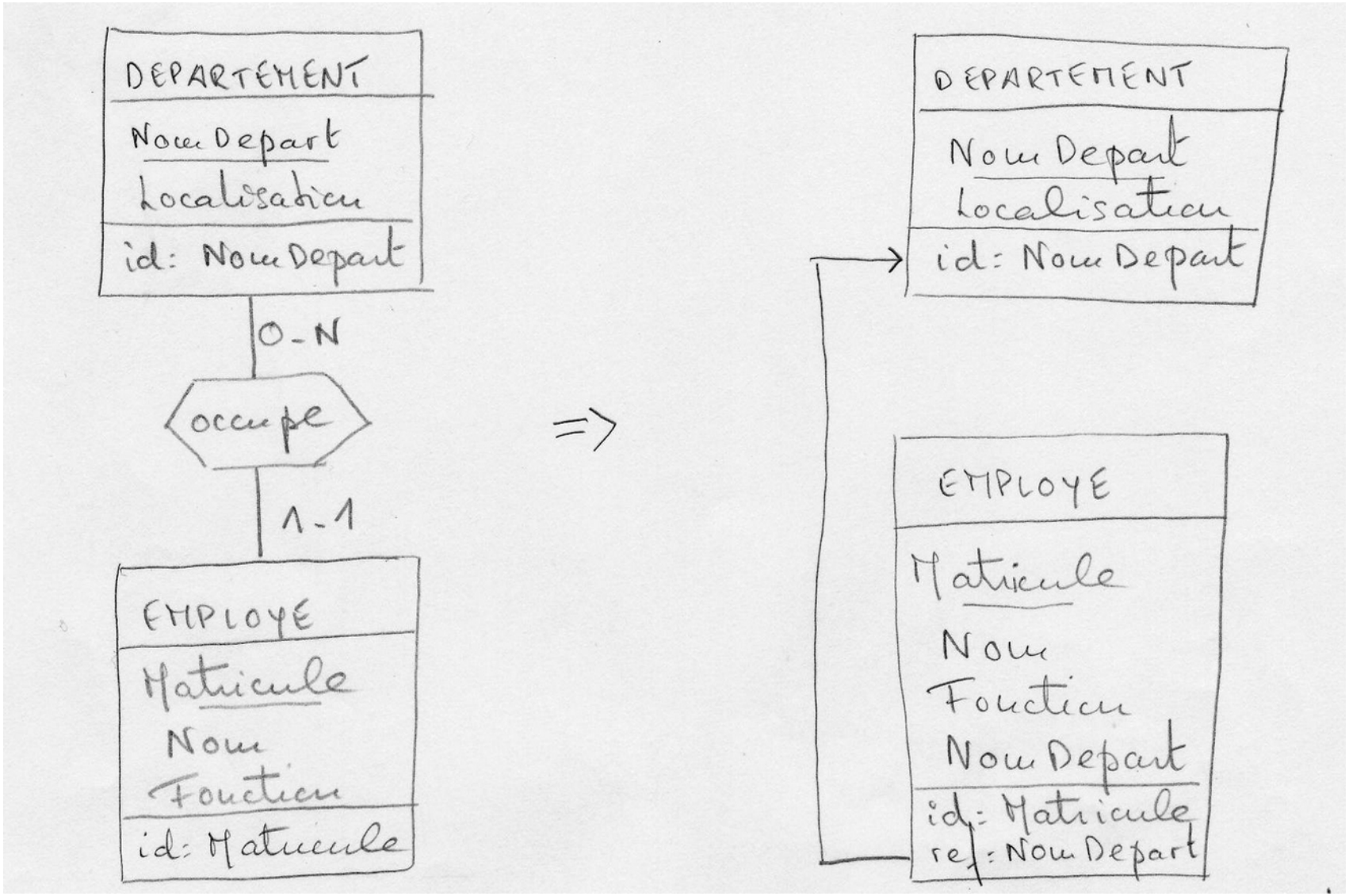
-> plusieurs entités B pour chaque entité A

- On représente R par une colonne RA de même type que IA, ajoutée à la table TB
- RA est une clé étrangère de B vers A
- RA conserve le caractère obligatoire ou facultatif de R dans B
- Cf. exemple

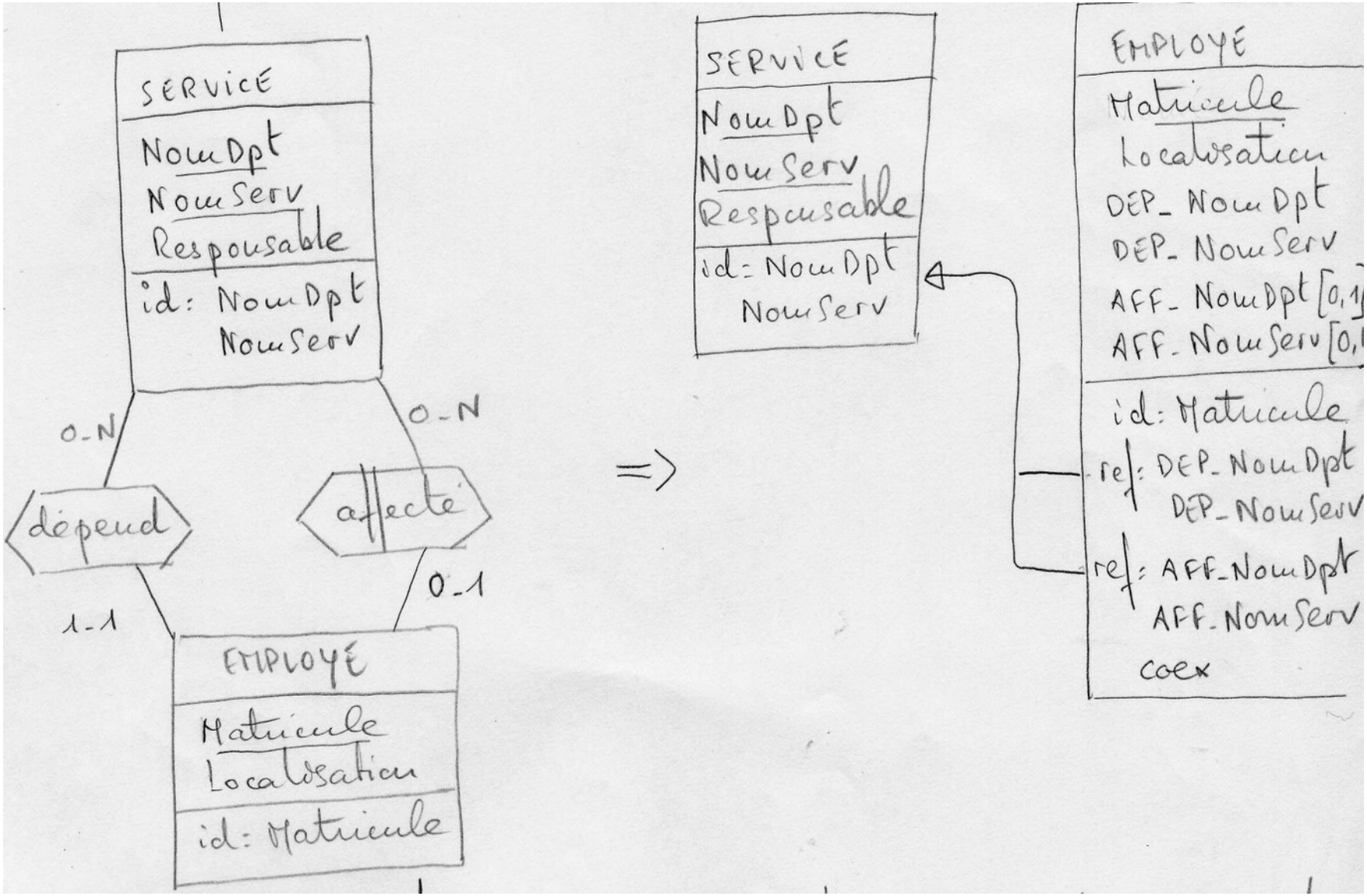
- Cas des clés étrangères multi-composants :
  - Si IA est constitué de plusieurs attributs IA_1,…, Ia_n on définit les colonnes RA_1,…, Ra_n dans TB
  - RA_1,…, Ra_n doivent être simultanément null ou simultanément non null (contrainte de coexistence)
  - Exemple

- Cas des identifiants hybrides
  - Cf. exemple

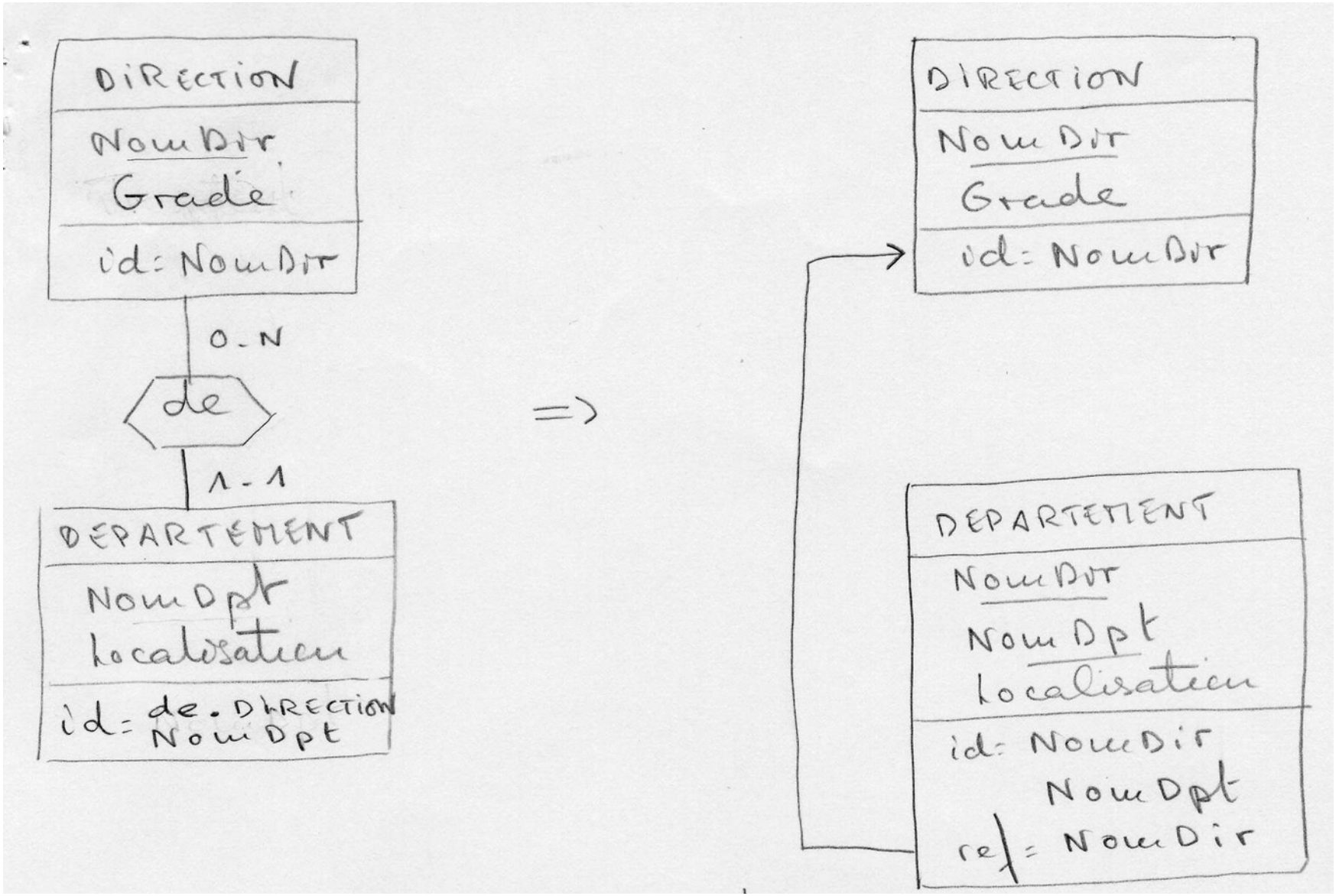
Exemple de représentation d'un type d'associations un-à-plusieurspar une clé étrangère



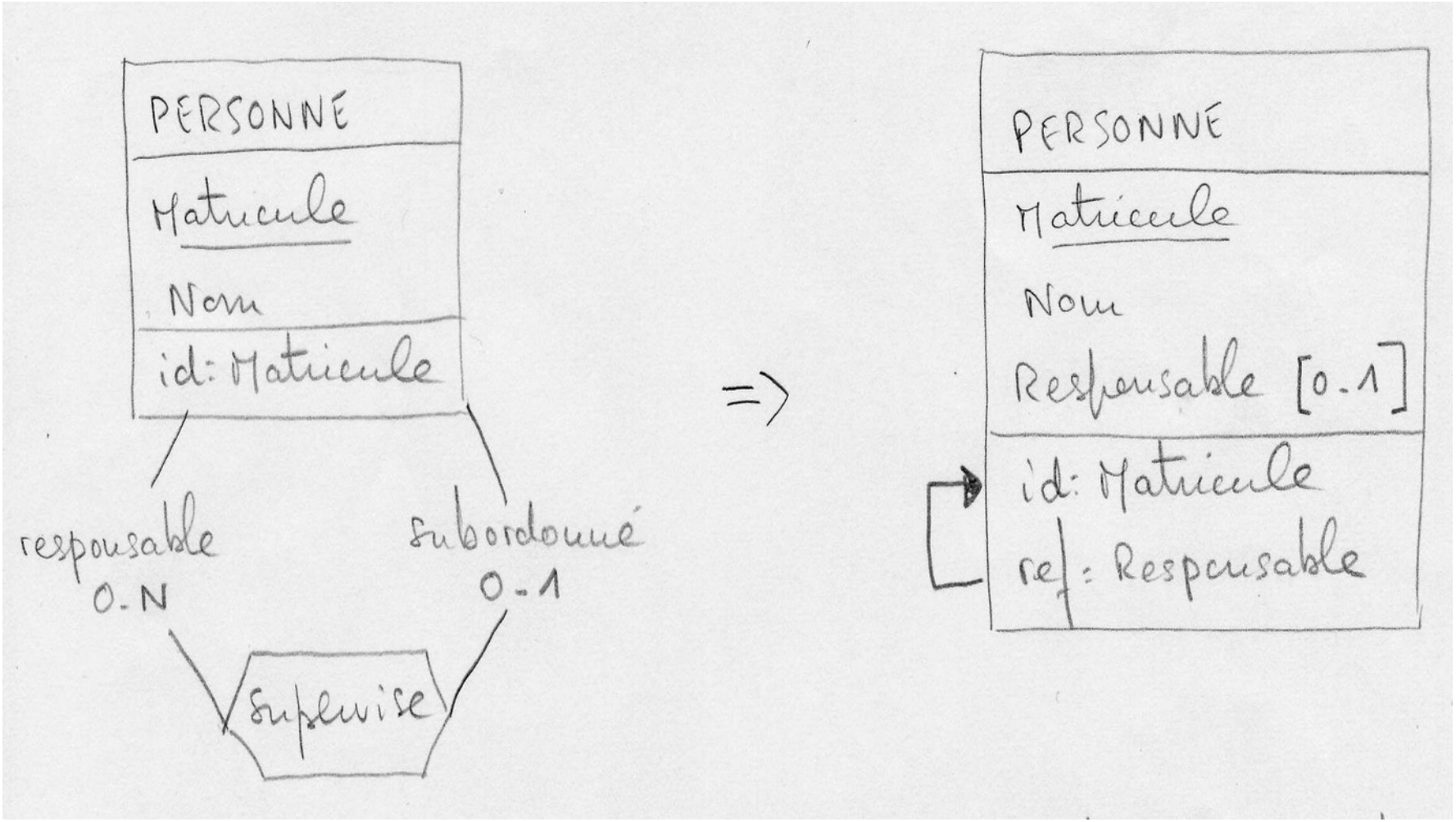
Exemple de traduction d'un type d'association un-à-plusieurs sous forme d'une clé étrangère multi-composant



Exemple de traduction d'un identifiant hybride



Exemple de représentation d'un type d'association un-à-plusieurs cyclique



4.2.2. Types d'associations un-à-un

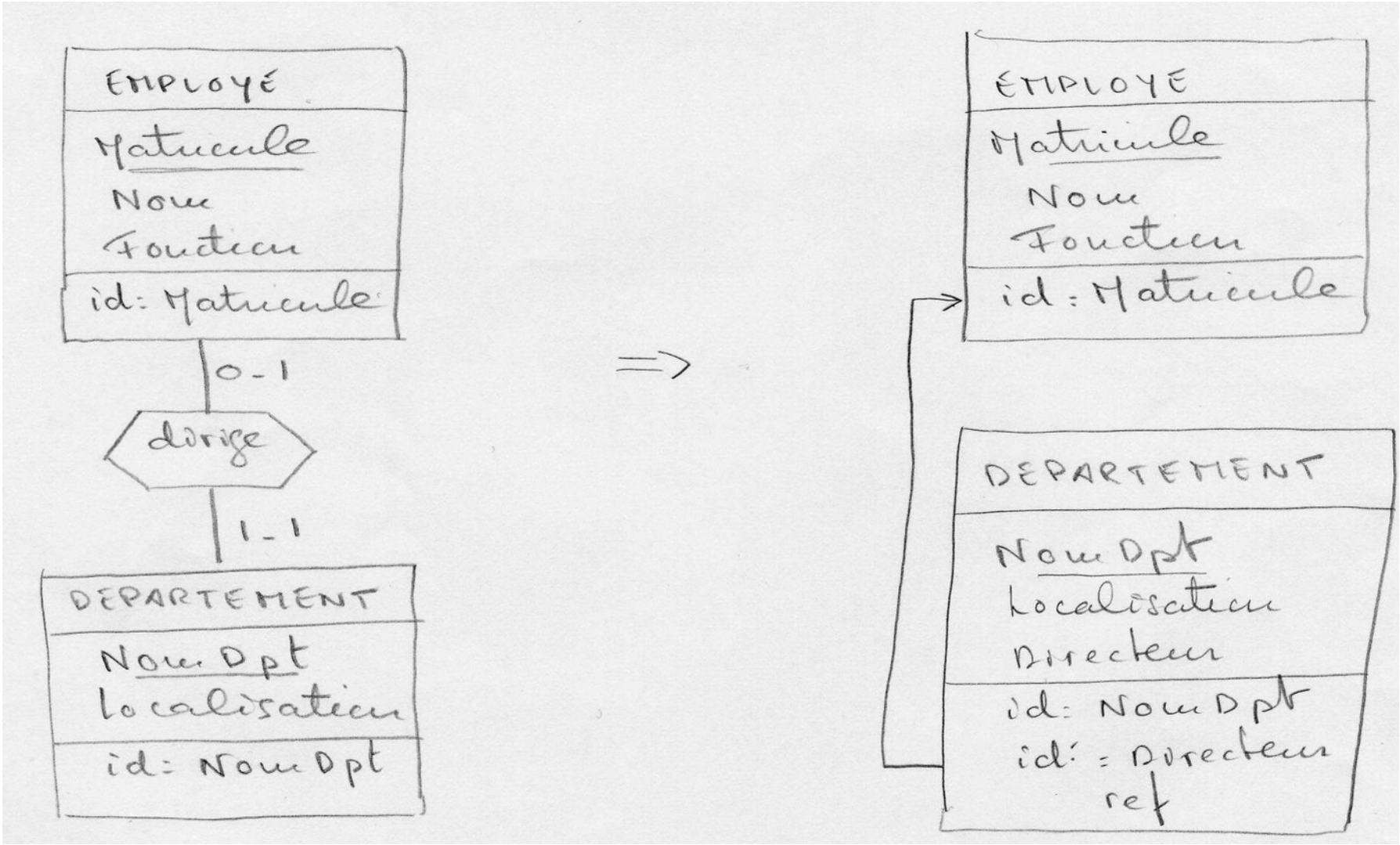
On représente R par l'ajout à la table TA (ou TB) d'une colonne RB (ou RA) de même type que l'identifiant IB (ou IA) de la table TB (ou TA), et on déclare cette colonne clé étrangère vers la table TB (ou TA).

- Si R est obligatoire pour A (resp. B) et facultatif pour B (resp. A), on ajoute une colonne RB (resp. RA) obligatoire dans TA (resp. TB)
- Si R est obligatoire pour A et B on choisit indifféremment l'une ou l'autre possibilité
- Si R est facultatif pour A et B la nouvelle colonne est déclarée facultative

Dans tous les cas la nouvelle colonne clé étrangère constitue un identifiant supplémentaire pour sa table

Cf. exemple

Exemple de représentation d'un type d'association un-à-un par une clé étrangère identifiante



4.2.3. Types d'associations plusieurs-à-plusieurs

- On transforme R en un type d'entités R' et deux types d'association un-à-plusieurs
- R' devient ensuite une table et les deux types d'association deviennent des clés étrangères

  Cf. exemple
- Exemple des types d'associations cycliques

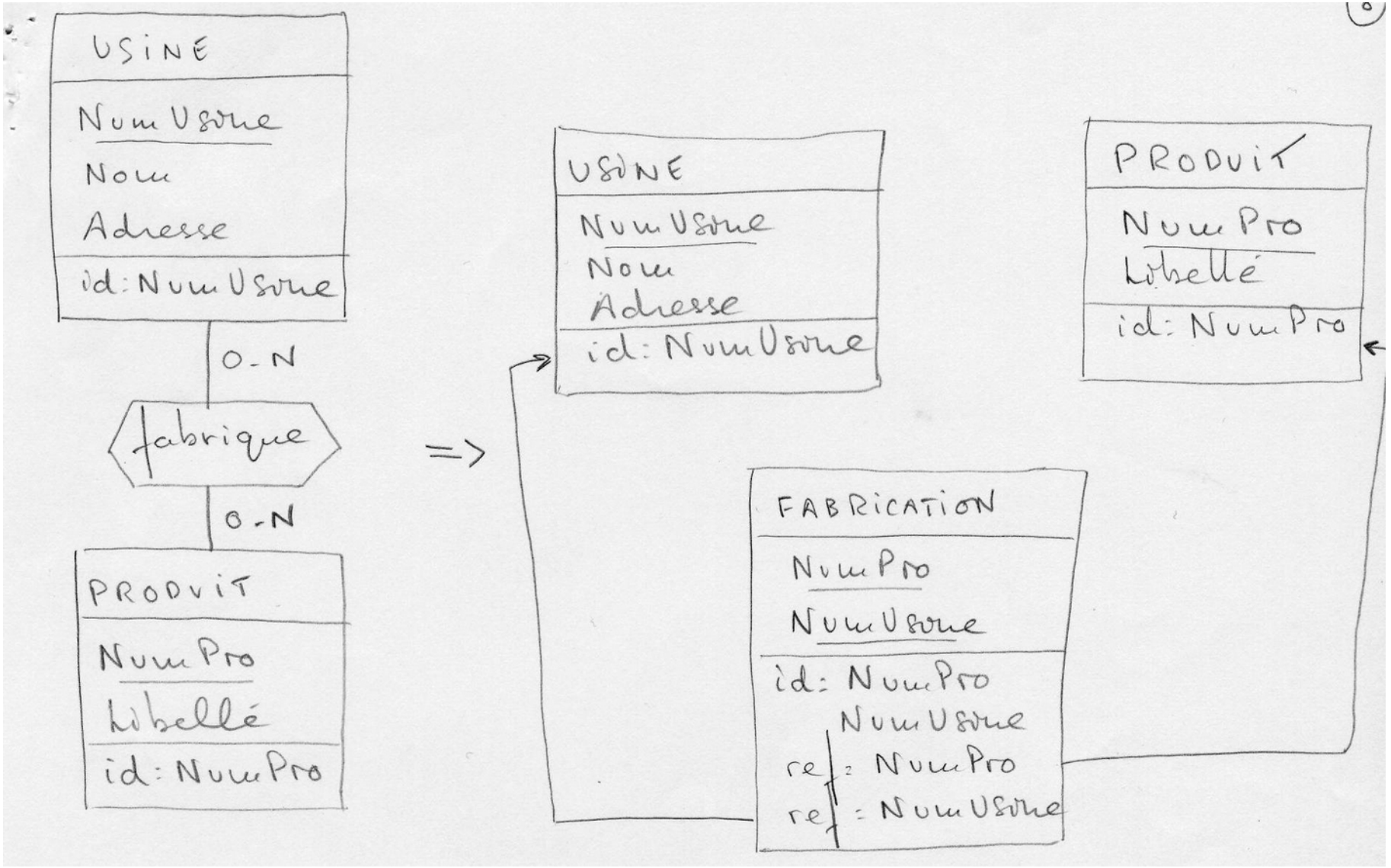
4.3. Synthèse graphique des règles de traduction

Cf. schémas

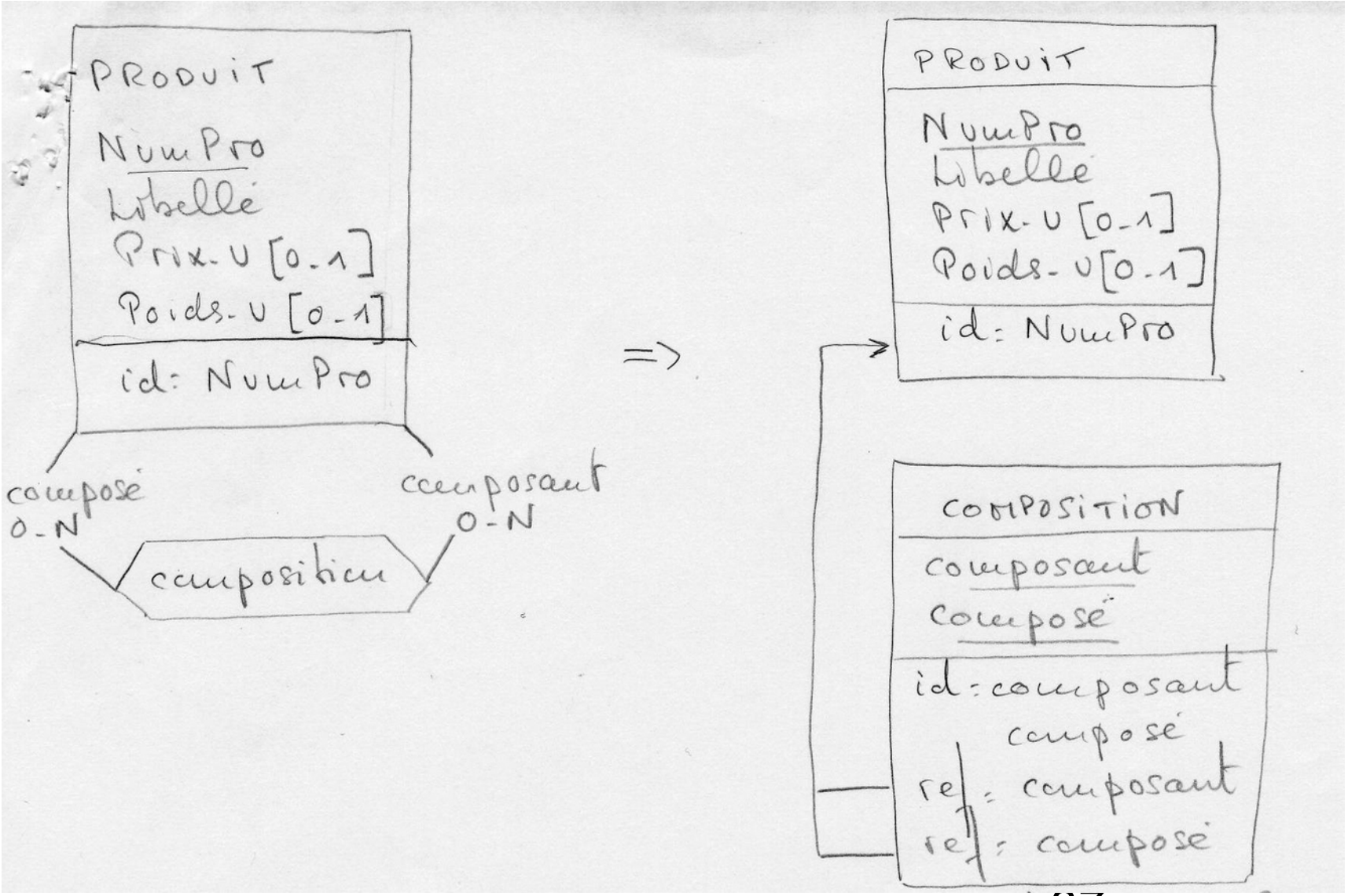
4.4. Traduction des structures en SQL

Exemple des contrats d'assurances automobiles

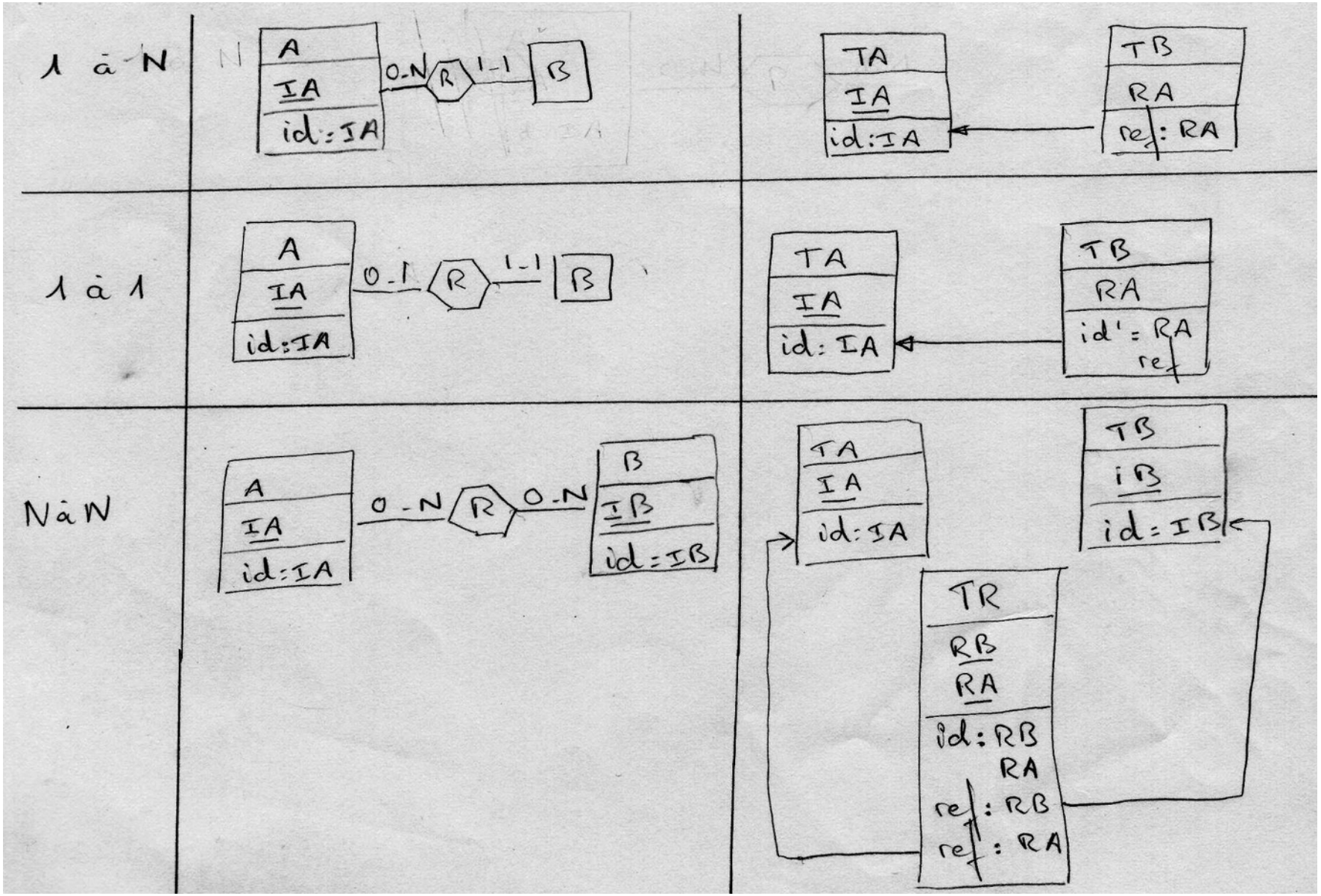
Exemple de représentation d'un type d'association plusieurs-à-plusieurs par une table associative



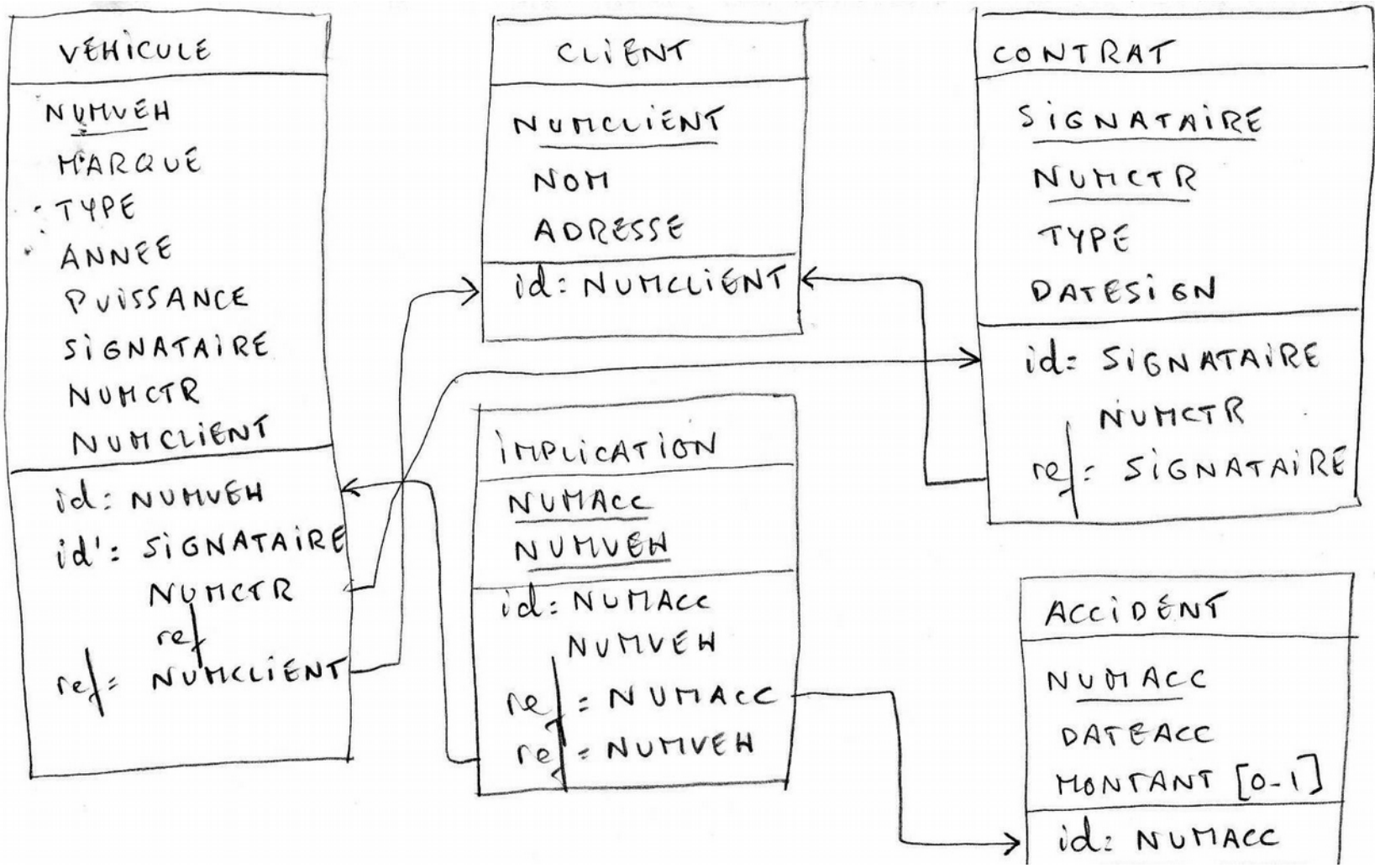
Exemple de représentation d'un type d'association plusieurs-à-plusieurs cyclique



4.4. Synthèse des règles de traduction des types d'associations



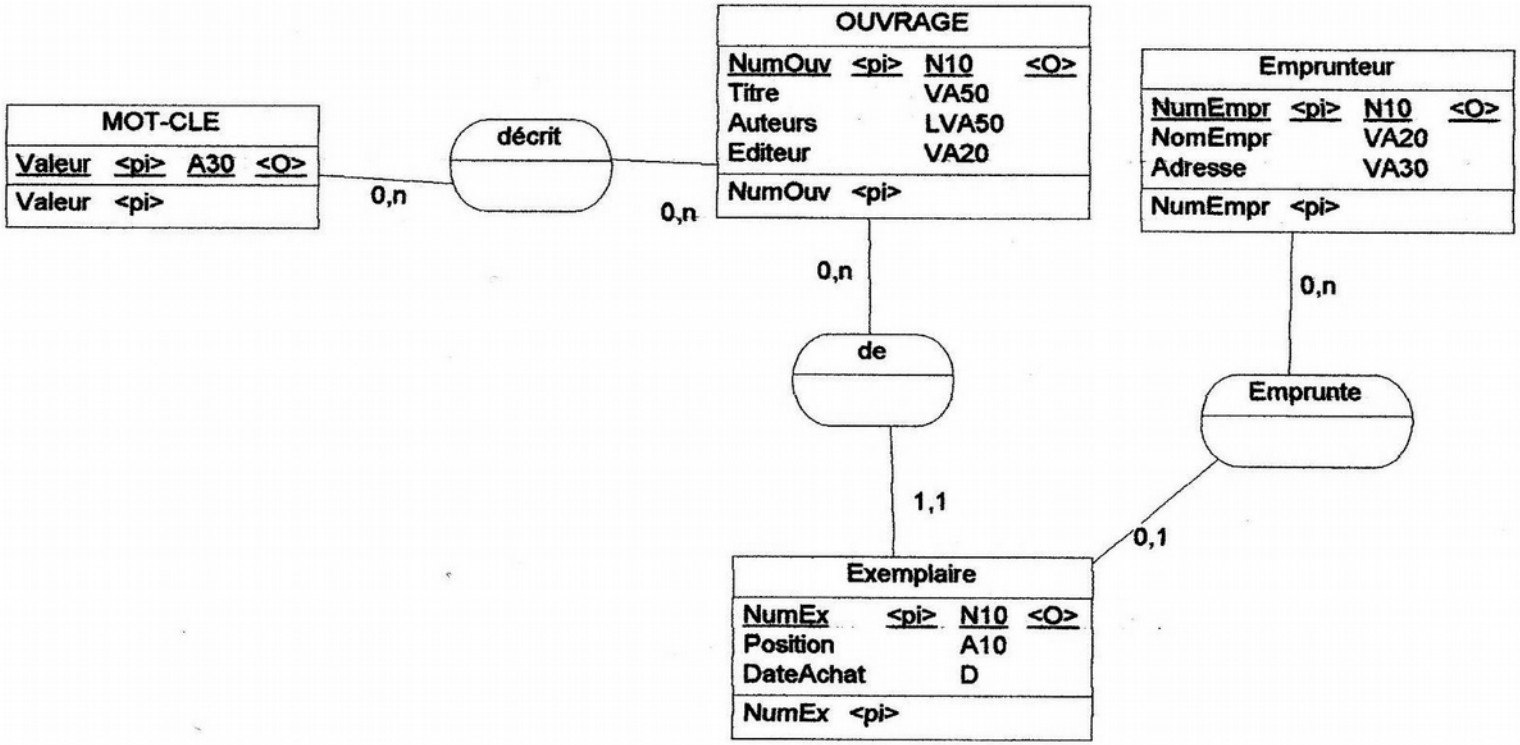
4.5. Exemple de traduction du schéma des assurances automobiles

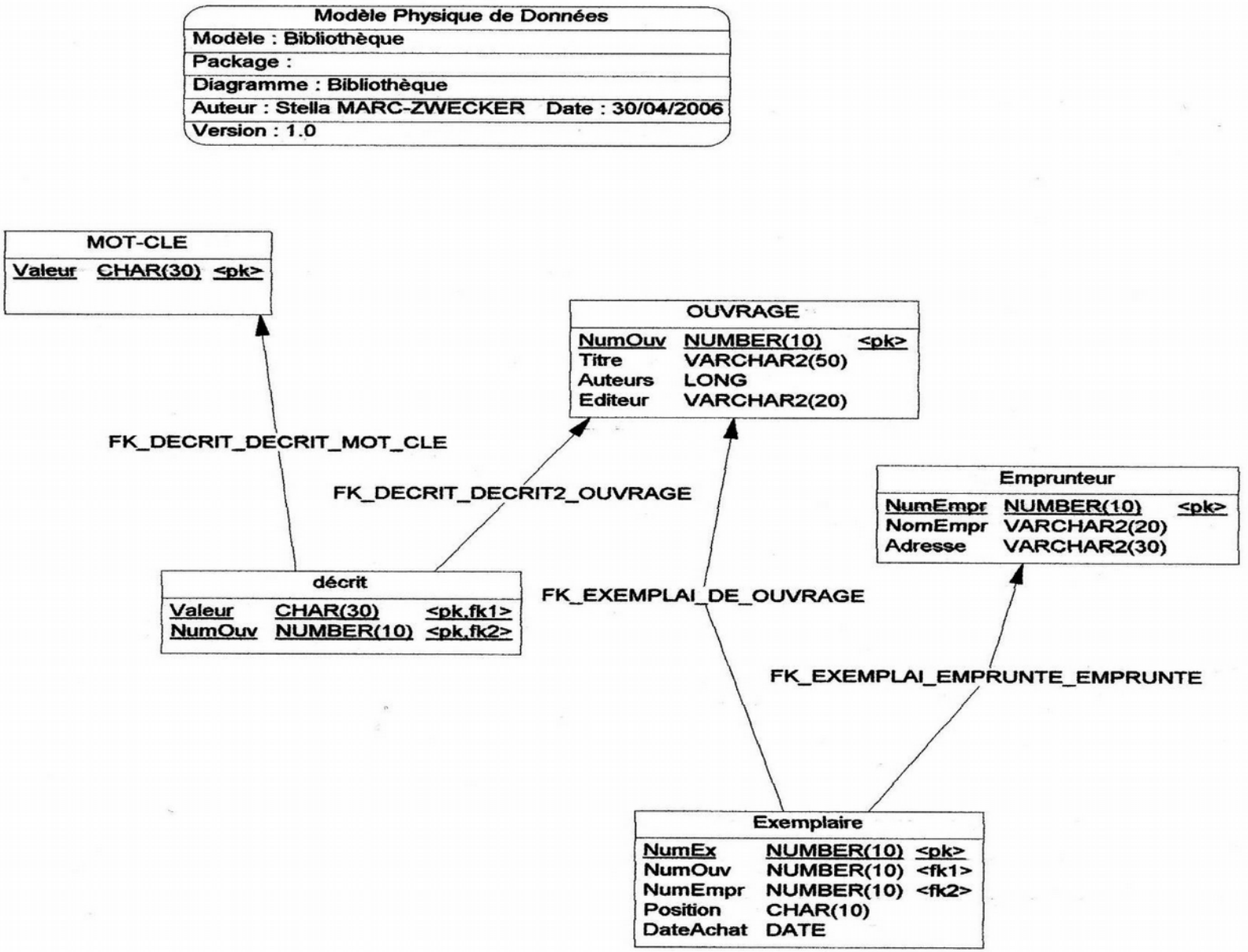


```
create table CLIENT ( NUMCLIENT char (12) not null,
                      NOM char (38) not null,
                      ADRESSE char (60) not null,
                    primary key (NUMCLIENT) );

create table CONTRAT ( SIGNATAIRE char (12) not null,
                       NUMCTR number (8) not null,
                       TYPE char (12) not null,
                       DATESIGN date not null,
                     primary key (SIGNATAIRE, NUMCTR),
                     foreign key (SIGNATAIRE) references CLIENT );

create table VEHICULE  ( NUMVEH char (16) not null,
                         MARQUE char (30) not null,
                         TYPE char (30) not null,
                         ANNEE number (4) not null,
                         PUISSANCE number (6) not null,
                         SIGNATAIRE char (12) not null,
                         NUMCTR number (8) not null,
                         NUMCLIENT char (12) not null,
                       primary key (NUMVEH),
                       unique (SIGNATAIRE, NUMCTR),
                       foreign key (NUMCLIENT) references CLIENT,
                       foreign key (SIGNATAIRE, NUMCTR) references CONTRAT);

create table ACCIDENT (NUMACC char (10) not null,
                       DATEACC date not null,
                       MONTANT number (6),
                     primary key (NUMACC));

create table IMPLICATION (NUMVEH char (16) not null,
                          NUMACC char (10) not null,
                       primary key (NUMVEH, NUMACC),
                       foreign key (NUMVEH) references VEHICULE,
                       foreign key (NUMACC) references ACCIDENT);
```

4.6. Exemple de la bibliothèque

Modèle Conceptuel de Données
Modèle : Bibliothèque
Package :
Diagramme : Bibliothèque
Auteur : Stella MARC-ZWECKER Date : 30/04/2006
Version : 1.0




Modèle Physique de Données
Modèle : Bibliothèque
Package :
Diagramme : Bibliothèque
Auteur : Stella MARC-ZWECKER Date : 30/04/2006
Version : 1.0
MOT-CLE
Valeur CHAR(30) <pk>
OUVRAGE
NumOuv NUMBER(10) <pk>
Titre VARCHAR2(50)
Auteurs LONG
Editeur VARCHAR2(20)
FK_DECRIT_DECRIT_MOT_CLE
FK_DECRIT_DECRIT2_OUVRAGE
Emprunteur
NumEmpr NUMBER(10) <pk>
NomEmpr VARCHAR2(20)
Adresse VARCHAR2(30)
décrit
Valeur CHAR(30) <pk,fk1>
NumOuv NUMBER(10) <pk,fk2>
FK_EXEMPLAI_DE_OUVRAGE
FK_EXEMPLAI_EMPRUNTE_EMPRUNTE
Exemplaire
NumEx NUMBER(10) <pk>
NumOuv NUMBER(10) <fk1>
NumEmpr NUMBER(10) <fk2>
Position CHAR(10)
DateAchat DATE

bibliotheque.sql

```
/*==============================================================*/
/* Nom de la base :  BIBLIOTHEQUE                               */
/* Nom de SGBD :  ORACLE Version 7                              */
/* Date de création :  30/04/2006 23:31:46                      */
/*==============================================================*/


alter table EXEMPLAIRE
   drop constraint FK_EXEMPLAI_EMPRUNTE_EMPRUNTE;

drop table EXEMPLAIRE cascade constraints;

drop table DECRIT cascade constraints;

drop table OUVRAGE cascade constraints;

drop table MOT_CLE cascade constraints;

drop table EMPRUNTEUR cascade constraints;

/*==============================================================*/
/* Table : EMPRUNTEUR                                           */
/*==============================================================*/

create table EMPRUNTEUR (
   NUMEMPR              NUMBER(10)                      not null,
   NOMEMPR              VARCHAR2(20),
   ADRESSE              VARCHAR2(30),
   constraint PK_EMPRUNTEUR primary key (NUMEMPR)
);

/*==============================================================*/
/* Table : MOT_CLE                                              */
/*==============================================================*/

create table MOT_CLE (
   VALEUR               CHAR(30)                        not null,
   constraint PK_MOT_CLE primary key (VALEUR)
);

/*==============================================================*/
/* Table : OUVRAGE                                              */
/*==============================================================*/

create table OUVRAGE (
   NUMOUV               NUMBER(10)                      not null,
   TITRE                VARCHAR2(50),
   AUTEURS              LONG,
   EDITEUR              VARCHAR2(20),
   constraint PK_OUVRAGE primary key (NUMOUV)
);
```

```
/*==============================================================*/
/* Table : DECRIT                                               */
/*==============================================================*/

create table DECRIT (
   VALEUR                 CHAR(30)                       not null,
   NUMOUV                 NUMBER(10)                     not null,
   constraint PK_DECRIT primary key (VALEUR, NUMOUV),
   constraint FK_DECRIT_DECRIT_MOT_CLE foreign key (VALEUR)
         references MOT_CLE (VALEUR),
   constraint FK_DECRIT_DECRIT2_OUVRAGE foreign key (NUMOUV)
         references OUVRAGE (NUMOUV)
);

/*==============================================================*/
/* Table : EXEMPLAIRE                                           */
/*==============================================================*/

create table EXEMPLAIRE (
   NUMEX                  NUMBER(10)                     not null,
   NUMOUV                 NUMBER(10)                     not null,
   NUMEMPR                NUMBER(10),
   POSITION               CHAR(10),
   DATEACHAT              DATE,
   constraint PK_EXEMPLAIRE primary key (NUMEX),
   constraint FK_EXEMPLAI_DE_OUVRAGE foreign key (NUMOUV)
         references OUVRAGE (NUMOUV),
   constraint FK_EXEMPLAI_EMPRUNTE_EMPRUNTE foreign key (NUMEMPR)
         references EMPRUNTEUR (NUMEMPR)
);
```

5. Etude de cas : les animaux du zoo [Hainaut]

5.1. Enoncé

- Les animaux du zoo suivent chacun un régime alimentaire
- Un régime est constitué d'un mélange d'ingrédients, chacun en quantité déterminée
- Le régime d'un animal peut varier d'un jour à l'autre
- Chaque animal est caractérisé, en fonction de son espèce, par ses besoins minima et maxima en nutriments (calcium, protéines, etc.) exprimés en mg par unité de poids de l'animal. Ces besoins sont fonction de l'espèce de l'animal
- On connaît la teneur de chaque ingrédient en nutriments exprimée en mg par kg d'ingrédient
- Chaque ingrédient a un coût unitaire
- Chaque animal requiert des soins qui sont évalués en euros par jour. Ces soins peuvent varier d'un jour à l'autre

5.2. Construction du schéma conceptuel

L'énoncé est décomposé en propositions élémentaires :

- *1. Un zoo a des animaux*

  => type d'entités ANIMAL

- *2. Un animal suit un régime*

  => type d'entités REGIME relié à ANIMAL par le type d'association *suit* (*un-à-un* : on considère qu'un régime est propre à un animal, bien que deux régimes puissent avoir la même composition)

- *3. Un régime est constitué d'ingrédients*

  => type d'entités INGREDIENT relié à REGIME par le type d'association *composé de* (*plusieurs-à-plusieurs* : un ingrédient peut entrer dans la composition de plusieurs régimes)

- *4. Un ingrédient entre dans la composition d'un régime en une quantité déterminée*

  => type d'entités COMPOSITION, qui recueille l'attribut « Quantité ». COMPOSITION est identifié par les types d'entités REGIME et INGREDIENT.

- *5. Le régime que reçoit un animal dépend du jour*

  => le type d'association *suit* est modifié pour devenir un-à-plusieurs. Un attribut « Date régime » est assigné au type d'entités REGIME

  => définition de l'identifiant hybride de REGIME (constitué de l'animal et de la date)

- *6. Un animal est d'une espèce*

  => on affecte à ANIMAL un attribut « Espèce »

- *7. Chaque animal est caractérisé, en fonction de son espèce, par ses besoins […] en nutriments. Ces besoins sont fonction de l'espèce de l'animal*
  - type d'entités NUTRIMENT qui doit être relié à *« Espèce » par un type d'associations*
  - *l'attribut « Espèce »* est transformé en type d'entités ESPECE, relié à ANIMAL par le type d'associations *origine* (de un-à-plusieurs)
  - type d'association *besoin de* (plusieurs-à-plusieurs) relie ESPECE et NUTRIMENT

- *8. Calcium, protéines sont des exemples de nutriments*
  - attribut « Nom nutriment » identifiant de NUTRIMENT

- *9. Les besoins d'une espèce en nutriments sont caractérisés par des quantités minimale et maximale*
  - transformation du type d'association *besoin de* en type d'entités BESOINS, caractérisé par les attributs « Min » et « Max »

- la valeur de « Min » doit être inférieure à celle de « Max » (définition de contrainte)

- *10. Chaque animal a un poids*
  - ANIMAL reçoit un attribut « Poids »

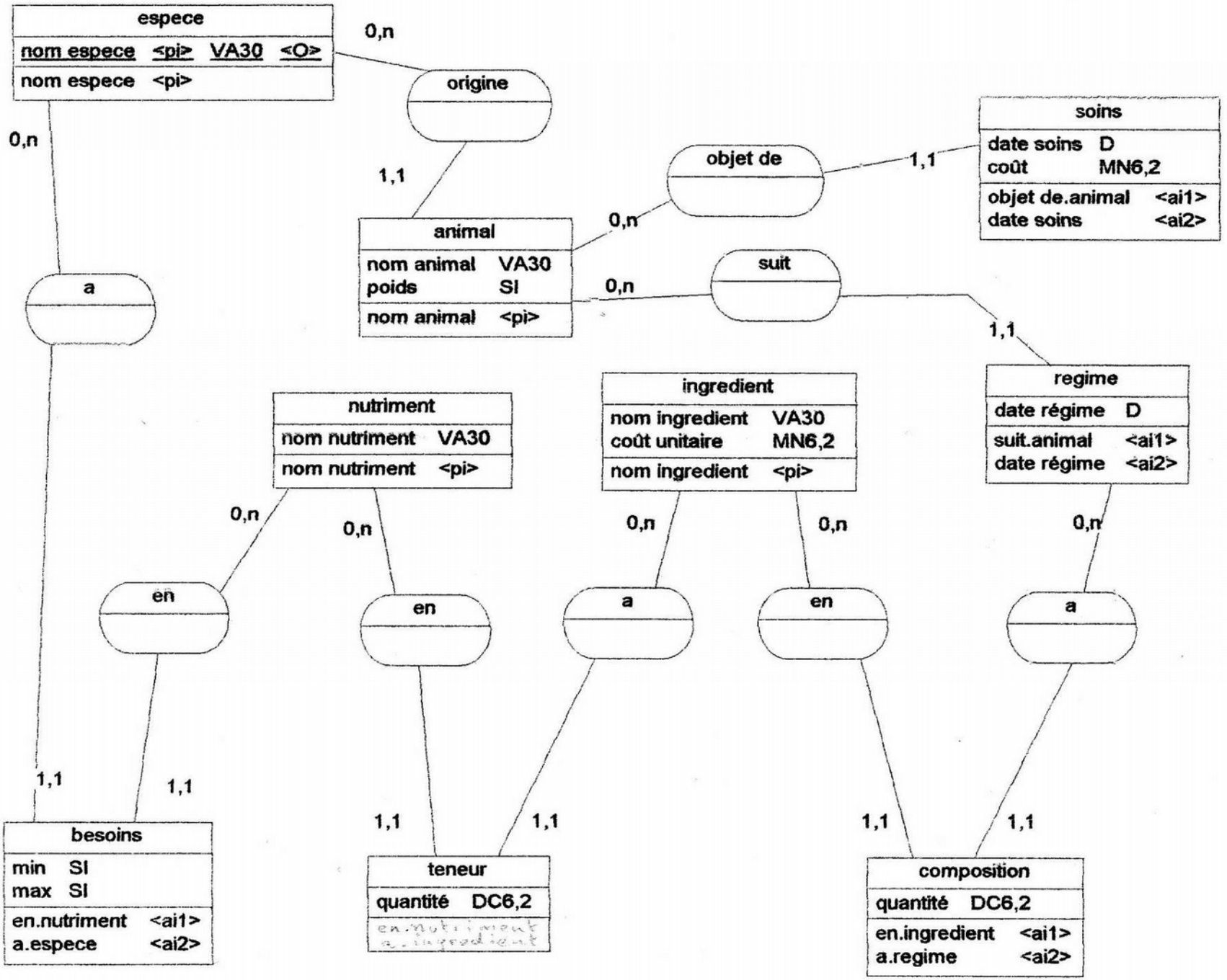
- *11. Chaque ingrédient a une teneur en nutriments, exprimée en mg/kg d'ingrédient*
  - une teneur est la quantité de nutriment contenue dans un ingrédient
  - type d'entités TENEUR, caractérisée par une quantité (en mg/kg), reliée à NUTRIMENT et à INGREDIENT

- *12. Chaque ingrédient a un coût unitaire*
  - INGREDIENT reçoit un attribut « Coût unitaire »

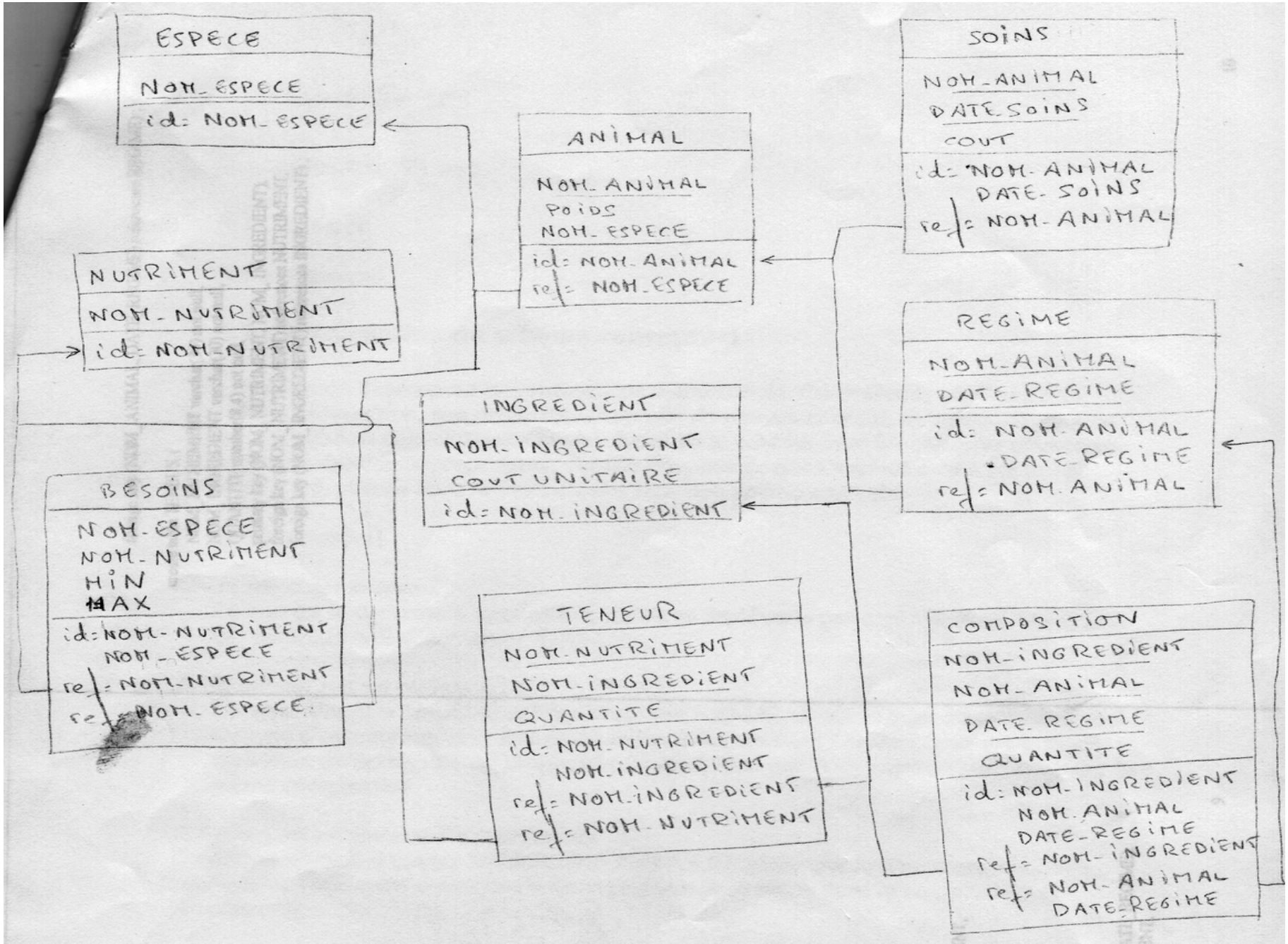
- *13. Chaque animal requiert des soins qui sont évalués en euros/jour*
  - ➔ type d'entités SOINS, caractérisé par un attribut « coût », relié à ANIMAL par un type d'associations *objet de* (un-à-plusieurs)

- *14. Les soins d'un animal dépendent du jour*
  - ➔ attribut « Date soins » qui constitue l'identifiant de SOINS avec l'animal objet des soins (identifiant hybride).

• Représentation graphique du schéma conceptuel du domaine d'application zoo (Cf. schéma)


espece
nom espece <pi> VA30 <O>
nom espece <pi>
0,n
origine
1,1
0,n
a
1,1
animal
nom animal VA30
poids SI
nom animal <pi>
objet de
1,1
soins
date soins D
coût MN6,2
objet de.animal <ai1>
date soins <ai2>
0,n
suit
1,1
regime
date régime D
suit.animal <ai1>
date régime <ai2>
nutriment
nom nutriment VA30
nom nutriment <pi>
ingredient
nom ingredient VA30
coût unitaire MN6,2
nom ingredient <pi>
0,n
0,n
0,n
0,n
0,n
en
en
a
en
a
1,1
1,1
1,1
1,1
1,1
1,1
besoins
min SI
max SI
en.nutriment <ai1>
a.espece <ai2>
teneur
quantité DC6,2
en.nutriment
a.ingredient
composition
quantité DC6,2
en.ingredient <ai1>
a.regime <ai2>

5.3. Production du schéma de tables



5.4. Production du code SQL

```
create table ESPECE (
     NOM_ ESPECE varchar(30) not null,
     primary key (NOM_ ESPECE)) ;

create table ANIMAL (
     NOM_ ANIMAL varchar(30) not null,
     POIDS number (6,2),
     NOM_ ESPECE varchar(30) not null,
     primary key (NOM_ ANIMAL)
foreign key (NOM_ ESPECE) references ESPECE) ;

create table INGREDIENT (
     NOM_INGREDIENT varchar(30) not null,
     COUT_UNITAIRE number (6,2) not null,
     primary key (NOM_INGREDIENT)) ;

create table NUTRIMENT (
     NOM_ NUTRIMENT varchar(30) not null,
     primary key (NOM_ NUTRIMENT)) ;
```

```
create table REGIME (
     NOM_ ANIMAL varchar(30) not null,
     DATE_ REGIME date not null,
     primary key (NOM_ ANIMAL, DATE_ REGIME),
        foreign key (NOM_ ANIMAL) references ANIMAL) ;

create table SOINS (
     NOM_ ANIMAL varchar(30) not null,
     DATE_ SOINS date not null,
     COUT number (6,2) not null,
     primary key (NOM_ ANIMAL, DATE_ SOINS),
        foreign key (NOM_ ANIMAL) references ANIMAL) ;

create table BESOINS (
     NOM_ ESPECE varchar(30) not null,
     NOM_ NUTRIMENT varchar(30) not null,
     MIN number (8,4) not null,
     MAX number (8,4) not null,
     primary key (NOM_ NUTRIMENT, NOM_ ESPECE),
        foreign key (NOM_ ESPECE) references ESPECE,
        foreign key (NOM_ NUTRIMENT) references NUTRIMENT,
        check (MAX >= MIN)) ;
```

```
create table COMPOSITION (
     NOM_ INGREDIENT varchar(30) not null,
     NOM_ ANIMAL varchar(30) not null,
     DATE_REGIME date not null,
     QUANTITE number(8,2) not null,
     primary key (NOM_ INGREDIENT, NOM_ ANIMAL, DATE_REGIME),
        foreign key (NOM_ INGREDIENT) references INGREDIENT,
        foreign key (NOM_ ANIMAL, DATE_REGIME) references REGIME) ;


create table TENEUR (
     NOM_ NUTRIMENT varchar(30) not null,
     NOM_ INGREDIENT varchar(30) not null,
     QUANTITE number(8,4) not null,
     primary key (NOM_ NUTRIMENT, NOM_ INGREDIENT),
        foreign key (NOM_ NUTRIMENT) references NUTRIMENT,
        foreign key (NOM_ INGREDIENT) references INGREDIENT) ;
```