Data Definition Language

Projet base de données – ENS Paris-Saclay

Guillaume Genestier
genestier@lsv.fr

15 février 2019

Motivation

- Les données d'une base de données sont organisées, conformément à un certain modèle logique de données.
- Parmi ces modèles logiques on s'intéressera exclusivement dans ce projet au modèle relationnel.
- Après l'élaboration du schéma conceptuel, il est nécessaire de traduire celui-ci en schéma en tables équivalent.

Modèle logique de données classique : le modèle relationnel.

- On s'intéresse à des types d'entités (appelés tables).
- Tout type d'entités possède au moins un attribut (appelé colonne).
- Un attribut est atomique ; il est obligatoire ou facultatif.
- Les seules contraintes sont celles qui sont induites par les identifiants (identifiant primaire ou secondaire) et les attributs de référence (clés étrangères).

Création d'une table

Syntaxe

```
create table CLIENT (
   NCLI       char(10),
   NOM        char(32),
   ADRESSE    char(60),
   LOCALITE   char(30),
   CAT        char(2),
   COMPTE     decimal(9,2)
)
```

Les Types

Quelques types utiles

Il s'agit de types relativement générique, cependant certains noms dépendent du SGBD utilisés (ici PostgreSQL est pris pour référence)

smallint	Entier signé court
int	Entier signé
bigint	Entier signé long
serial	Entier à incrémentation automatique
decimal(p,s)	Nombre décimal de p chiffres dont s après la virgule
float	Nombre à virgule flottante
varchar(p)	Chaîne de caractères de longueur inférieure à p
bool	Booléen
date	Date (année, mois et jour)
time	Instants (heure, minute, seconde)
timestamp	Date et heure

Créer ses propres types

Nommer un type

```
create domain MONTANT decimal(9,2)
```

Affecter une valeur par défaut

```
create domain MONTANT decimal(9,2) default 0.0
```

Créer un type somme

```
create type HUMEUR as enum('gai','triste')
```

Création d'une table

Unicité

On déclare une clé primaire par le mot clé `primary key`.
On déclare un attribut comme étant unique par `unique`.

Champ obligatoire

On déclare un attribut comme obligatoire par le mot clé `not NULL`.

On combine tout ça

```
create table CLIENT (
  NCLI      char(10),
  NOM       char(32) not NULL,
  ADRESSE   char(60),
  LOCALITE  char(30),
  CAT       char(2) default 'AA',
  unique (NOM,ADRESSE,LOCALITE),
  primary key (NCLI)
)
```

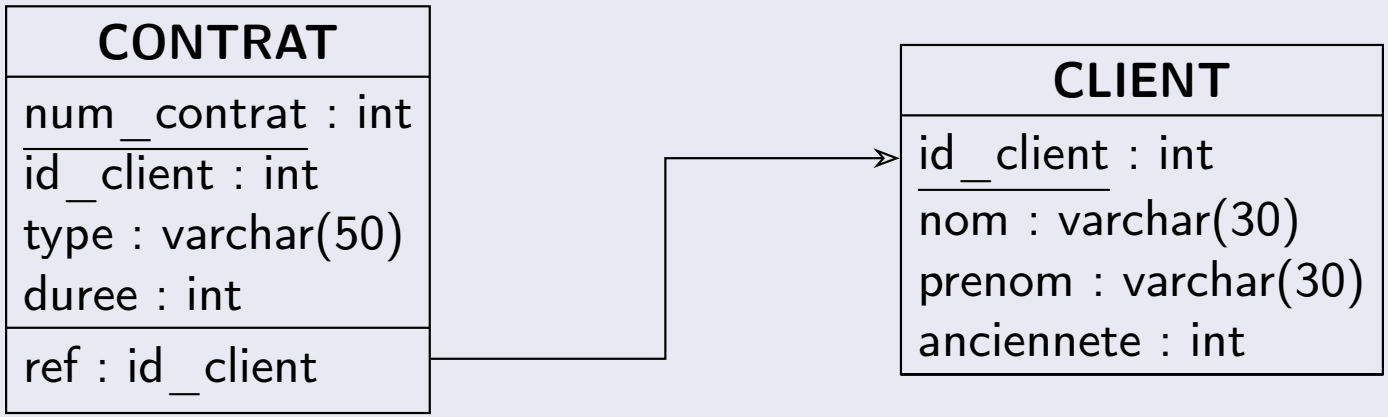
Clés étrangères

Définitions

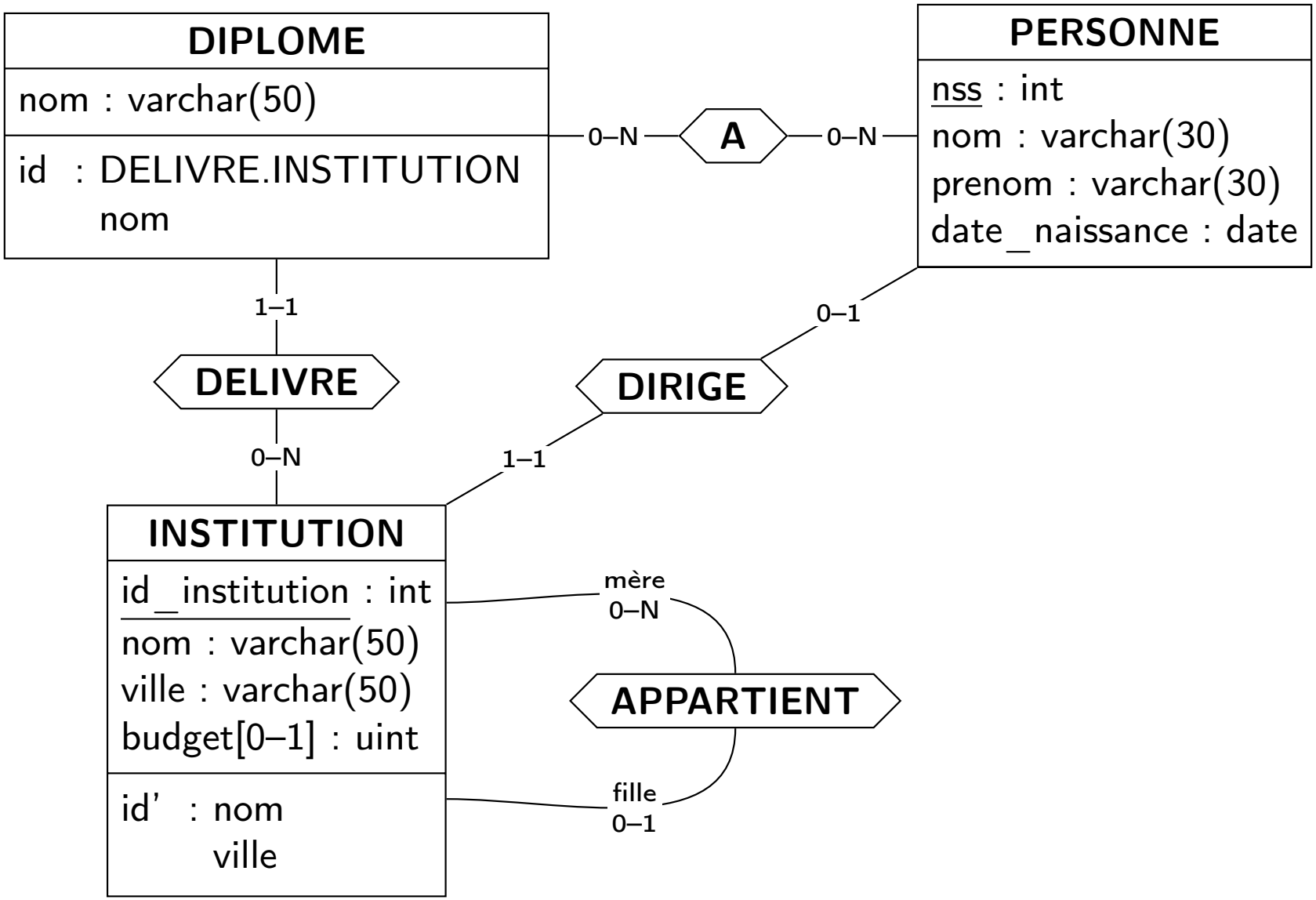
- Étant donné deux tables T et T', une clé étrangère de T vers T' est un ensemble de colonnes E de T, copie d'un identifiant E' de T' vérifiant que pour tout n-uplet t de T, il existe un n-uplet t' de T' tel que la valuation donnée aux colonnes de E dans t corresponde à la valuation donnée aux colonnes de E' dans t' (contrainte référentielle).
- Les colonnes de E et E' sont donc en même nombre et de mêmes types.
- Les colonnes de E sont soit toutes obligatoires, soit toutes facultatives et coexistantes (tout n-uplet a soit toutes ses colonnes à NULL, soit elles sont toutes différentes de NULL et référencent un n-uplet de T').
- En général, E' est l'identifiant primaire de T'.

Clés étrangères

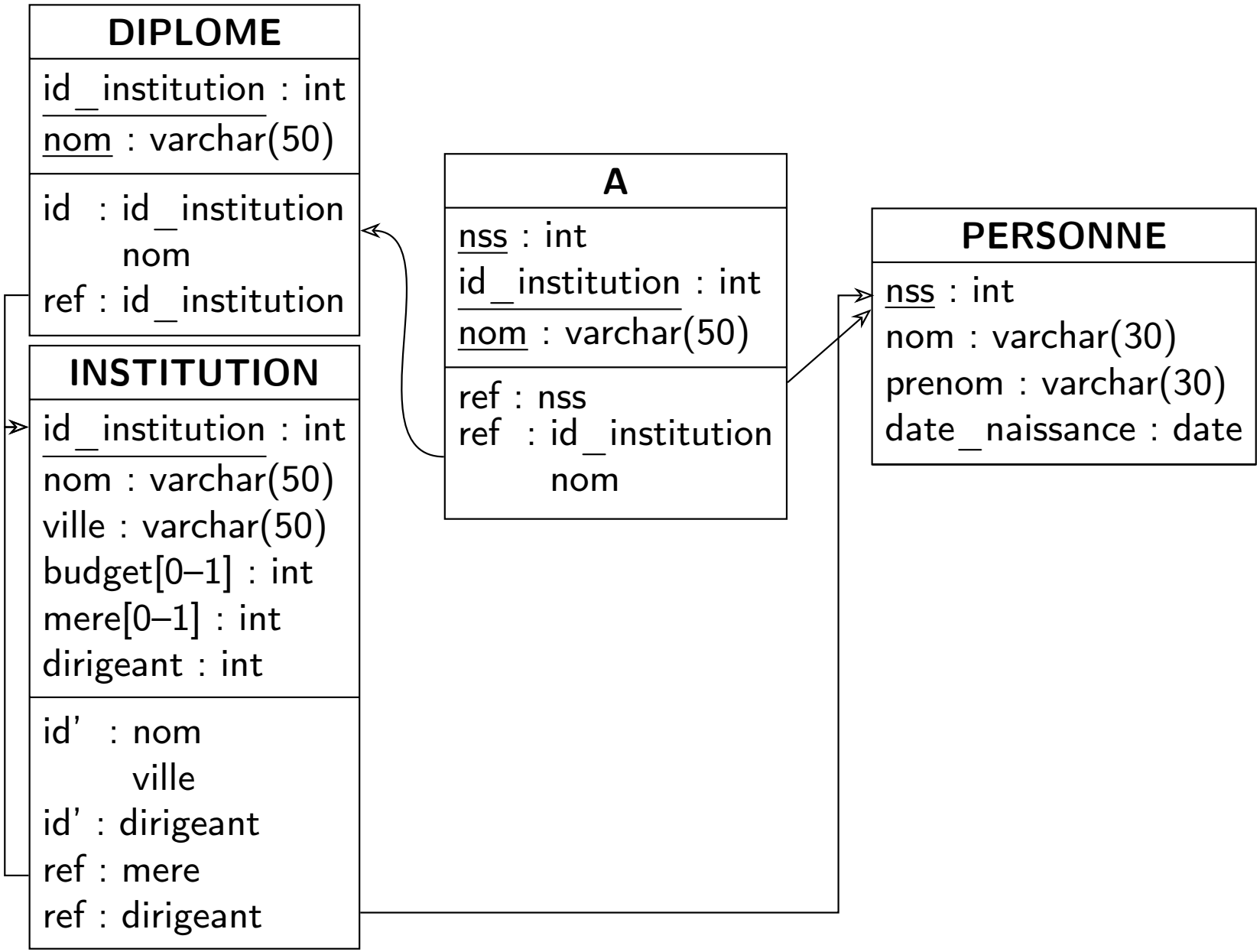
Exemple



Traduction


DIPLOME
nom : varchar(50)
id : DELIVRE.INSTITUTION
nom
0–N
A
0–N
PERSONNE
nss : int
nom : varchar(30)
prenom : varchar(30)
date_naissance : date
1–1
DELIVRE
0–N
0–1
DIRIGE
1–1
INSTITUTION
id_institution : int
nom : varchar(50)
ville : varchar(50)
budget[0–1] : uint
id' : nom
ville
mère
0–N
APPARTIENT
fille
0–1

Traduction


DIPLOME
id_institution : int
nom : varchar(50)
id : id_institution
nom
ref : id_institution
INSTITUTION
id_institution : int
nom : varchar(50)
ville : varchar(50)
budget[0–1] : int
mere[0–1] : int
dirigeant : int
id' : nom
ville
id' : dirigeant
ref : mere
ref : dirigeant
A
nss : int
id_institution : int
nom : varchar(50)
ref : nss
ref : id_institution
nom
PERSONNE
nss : int
nom : varchar(30)
prenom : varchar(30)
date_naissance : date

Création d'une table

Clé étrangère

On déclare une clé étrangère par le mot clé `foreign key` et la table référencée est désignée en utilsant `references`.

Exemple

```
create table DETAIL (
 NCOM char(12) not null,
 NPRO char(15) not null,
 QCOM int      not null,
 primary key (NCOM,NPRO),
 foreign key (NCOM) references COMMANDE,
 foreign key (NPRO) references PRODUIT
 )
```

Pas de panique, rien n'est gravé dans le marbre

```
drop table DETAIL

alter table PRODUIT add column POIDS smallint

alter table PRODUIT drop column PRIX

alter table CLIENT alter column CAT set '00'

alter table CLIENT add primary key (NCLI)

alter table CLIENT add unique (NOM,ADRESSE,LOCALITE)

alter table CLIENT modify CAT not null

alter table CLIENT modify ADRESSE null

alter table COMMANDE
add foreign key (NCLI) references CLIENT
```

Nommer les contraintes

On peut nommer les contraintes avec le mot clé `constraint` suivi du nom que l'on veut lui donner.

Exemple

```
create table DETAIL (
 NCOM char(12) not null,
 NPRO char(15) constraint C1 not null,
 QCOM int      not null,
 constraint C2 primary key (NCOM,NPRO),
 constraint C3 foreign key (NCOM)
               references COMMANDE,
 foreign key (NPRO) references PRODUIT
 )
```

C'est utile pour les faire évoluer

```
alter table DETAIL drop constraint C2
```