

# Modélisation et conception de bases de données

L3Pro SCT – Bases de données et programmation

Mathieu Sassolas

IUT de Sénart Fontainebleau  
Département Informatique

Année 2015-2016  
Cours 2



Schémas

M. Sassolas  
L3Pro SCT – M7

Cours 2

Conception

Implémentation

TD/TP

1 Concevoir un schéma de base de donnée

2 Créer les table en SQL

3 TD/TP

2 / 24



Schémas

M. Sassolas  
L3Pro SCT – M7

Cours 2

Conception

Implémentation

TD/TP

1 Concevoir un schéma de base de donnée

2 Créer les table en SQL

3 TD/TP

3 / 24



Schémas

M. Sassolas  
L3Pro SCT – M7

Cours 2

Conception

Implémentation

TD/TP

- ▶ Le schéma de la base, c'est à dire de quelles tables avons nous besoin.
- ▶ Le schéma de chaque table, c'est à dire :
  - combien de colonnes ;
  - le nom de ces colonnes ;
  - le type des données qu'elles contiennent.
- ▶ Les liens entre les tables : **clef référencées**.

4 / 24



## Comment le représenter ?

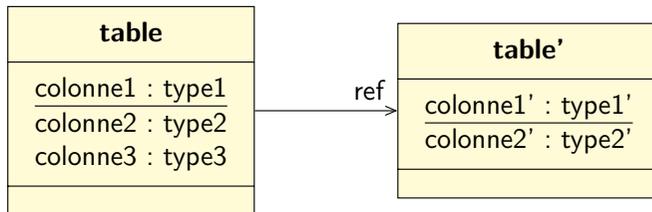
Schémas

M. Sassolas  
L3Pro SCT - M7  
Cours 2

Conception

Implémentation  
TD/TP

On utilise des diagrammes de classes d'UML.



### Quelques remarques sur la syntaxe

- ▶ Colonne soulignée : **clef primaire**.
- ▶ Flèche : **référence** d'une table à l'autre.
- ▶ Lien : référence dans les deux sens.
- ▶ Les noms sur les liens/flèches donnent le nom de la colonne dans la table de l'**autre extrémité**.
- ▶ On pointe toujours vers la **clef primaire**.

5 / 24



## Quelles tables ?

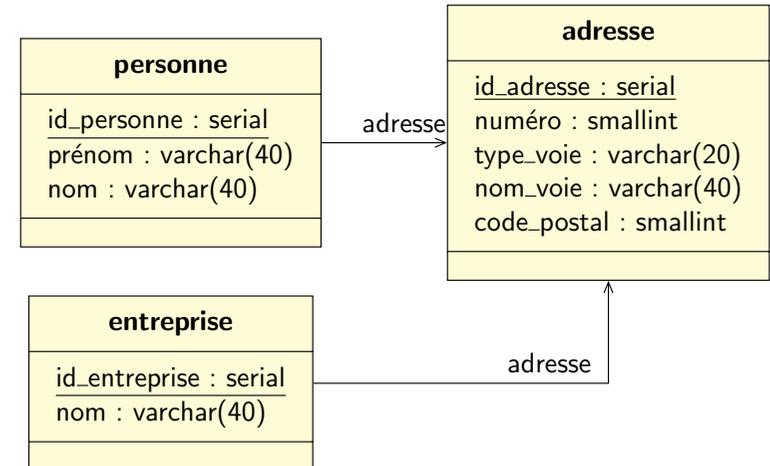
Schémas

M. Sassolas  
L3Pro SCT - M7  
Cours 2

Conception

Implémentation  
TD/TP

- ▶ Une table va garder une seule sorte d'objets.
- ▶ Si des parties de l'objet sont utilisées ailleurs, on peut en faire une autre table et la référencer.



6 / 24



## Quelles colonnes ?

Schémas

M. Sassolas  
L3Pro SCT - M7  
Cours 2

Conception

Implémentation  
TD/TP

- ▶ Toutes les informations, découpées selon les types de base de SQL, c'est à dire :
  - des chaînes de caractères,
  - des nombres entiers,
  - des nombres à virgule (€, \$, ...),
  - des nombres à virgule flottante,
  - des dates,
  - des booléens,
  - des données binaires.
- ▶ Les informations plus complexes sont présentes par référence uniquement.

### Lors de la conception

On n'est pas obligé de donner plus de détails sur les types.

7 / 24



## Associations entre les tables

Schémas

M. Sassolas  
L3Pro SCT - M7  
Cours 2

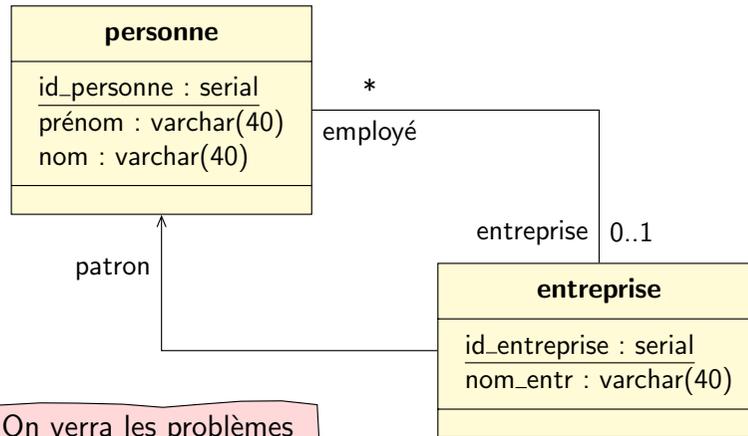
Conception

Implémentation  
TD/TP

- ▶ Explicite les références.
- ▶ Peuvent être multidirectionnels (lien) ou unidirectionnels (flèche).
- ▶ Peuvent avoir une **cardinalité** : désigne le nombre d'éléments qui peuvent participer à ce bout de l'association.
  - 1 (par défaut)
  - 0..1 : optionnel
  - 0..\* ou \* : un nombre arbitraire.
  - 1..\* : au moins 1

8 / 24





On verra les problèmes des cardinalités \* lors de l'implémentation

- 1 Concevoir un schéma de base de donnée
- 2 Créer les table en SQL
- 3 TD/TP

1. Donner des types SQL aux types « abstraits » (entier, chaîne de caractères, ...); attention, les types peuvent dépendre du système de base de donnée : on prend les types de PostgreSQL.
2. Préparer chaque table individuellement.
3. Implémenter les liens entre les tables par des **contraintes référentielles**.

Il y a plusieurs types SQL pour chaque type abstrait.

Type abstrait	Types PostgreSQL
chaînes de caractères	text, character(n), char(n), character varying(n), varchar(n)
entier	smallint, integer, bigint
entier auto-incrémenté	smallserial, serial, bigserial
nombres à virgule exacte	numeric, decimal, money
nombres à virgule flottante	real, double precision
date	date, time without timezone, timestamp without timezone, time with timezone, timestamp with timezone
booléen	boolean
données binaires	bytea

### Syntaxe

```
CREATE TABLE <nom_table> (
  <colonne1> <type1> <options>,
  <colonne2> <type2> <options>,
  :
);
```

### Exemple

```
CREATE TABLE personne (
  id_personne serial PRIMARY KEY,
  pseudo varchar(40) UNIQUE,
  nom_personne varchar(40) NOT NULL,
  prenom_personne varchar(40) NOT NULL,
  date_naissance date,
  active boolean DEFAULT TRUE);
```

**UNIQUE** Un seul tuple dispose de cette valeur dans cette colonne.

**NOT NULL** Valeur NULL non acceptée.

**PRIMARY KEY** Cette colonne est la clef primaire ; implique UNIQUE NOT NULL.

**DEFAULT <valeur>** Donne une valeur par défaut.

### Remarque sur les clefs primaires

- ▶ Une clef primaire peut correspondre à plusieurs colonnes conjointement. On peut le spécifier à la fin, après les colonnes :  
PRIMARY KEY(<colonne1>,<colonne2>,...)
- ▶ On peut faire de même avec UNIQUE.

Un journal a une date de parution, un volume (en général un par an), un numéro, et correspond à une publication (qui elle contiendra entre autres le titre du journal. La paire (volume,numéro) est la clef primaire.

```
CREATE TABLE journal (
  volume integer NOT NULL,
  numero integer NOT NULL,
  parution date NOT NULL
  DEFAULT CURRENT_TIMESTAMP,
  publication integer, -- Sera une référence
  PRIMARY KEY(volume,numero)
  UNIQUE (publication,parution)
);
```

### Syntaxe

```
ALTER TABLE <table>
  ADD COLUMN <colonne> <type> <options>;
```

```
ALTER TABLE <table> DELETE COLUMN <colonne>;
```

```
DROP TABLE <table>;
```

### Exemple

```
ALTER TABLE personne DELETE COLUMN date_naissance;
ALTER TABLE personne ADD COLUMN taille numeric;
DROP TABLE table_inutile;
```

## Ajouter des références

Schémas

M. Sassolas  
L3Pro SCT - M7

Cours 2

Conception

Implémentation

TD/TP

↔ Il faut avoir déjà créé les deux tables.

### Syntaxe

```
ALTER TABLE <table> ADD FOREIGN KEY (<colonne>)
REFERENCES <table_référencée>;
```

```
ALTER TABLE <table> ADD FOREIGN KEY (<colonne>)
REFERENCES <table_référencée>(<colonne_référencée>;
```

```
ALTER TABLE <table> ADD CONSTRAINT <nom_contrainte>
FOREIGN KEY (<colonne>) REFERENCES <table_référencée>;
```

### Remarques

- ▶ Par défaut la colonne référencée est la clef primaire.
- ▶ Sans nom de contrainte, le système donne un nom standard tout à fait explicite.

17 / 24



## Exemple de création de références

Schémas

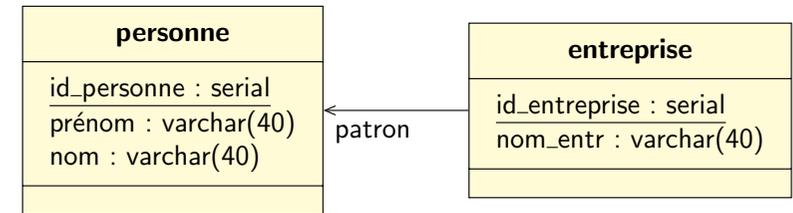
M. Sassolas  
L3Pro SCT - M7

Cours 2

Conception

Implémentation

TD/TP



```
CREATE TABLE personne (id_personne serial PRIMARY KEY,
prénom varchar(40) NOT NULL, nom varchar(40) NOT
NULL);
```

```
CREATE TABLE entreprise (id_entreprise serial PRIMARY
KEY, nom_entr varchar(40) NOT NULL);
```

```
ALTER TABLE entreprise ADD FOREIGN KEY (patron)
REFERENCES personne;
```

18 / 24



## Aveu : on peut gagner du temps

Schémas

M. Sassolas  
L3Pro SCT - M7

Cours 2

Conception

Implémentation

TD/TP

- ▶ Si la table référencée existe déjà, on peut directement créer une référence.
- ▶ Syntaxe : CREATE TABLE deuxième ( colonne1 serial PRIMARY KEY, colonne2 integer REFERENCES première );
- ▶ Il est donc utile de réfléchir à l'ordre dans lequel on crée les tables.

19 / 24



## Références avec cardinalité multiple

Schémas

M. Sassolas  
L3Pro SCT - M7

Cours 2

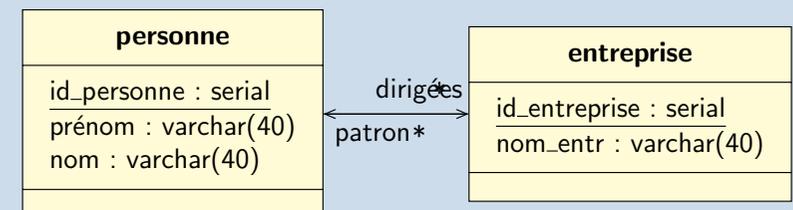
Conception

Implémentation

TD/TP

- ▶ Les cardinalités multiple en source de référence ne posent pas de problème.
- ▶ Il est parfois possible d'« inverser » le sens d'une référence pour que la cardinalité multiple soit à la source ; on peut alors retrouver les valeurs des sources par un SELECT.

### Exemple

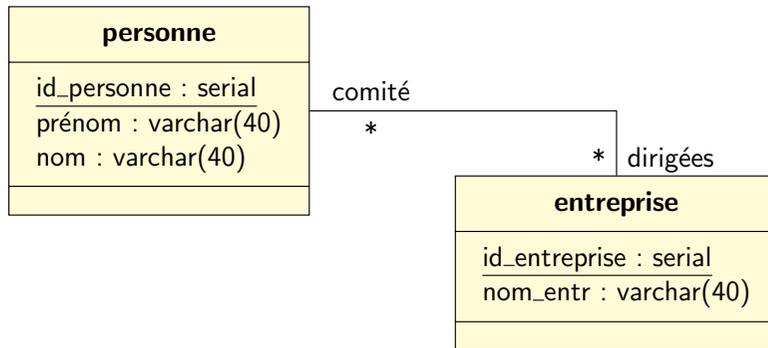


```
SELECT id_entreprise FROM entreprise WHERE
patron=<valeur_id_personne>;
```

20 / 24

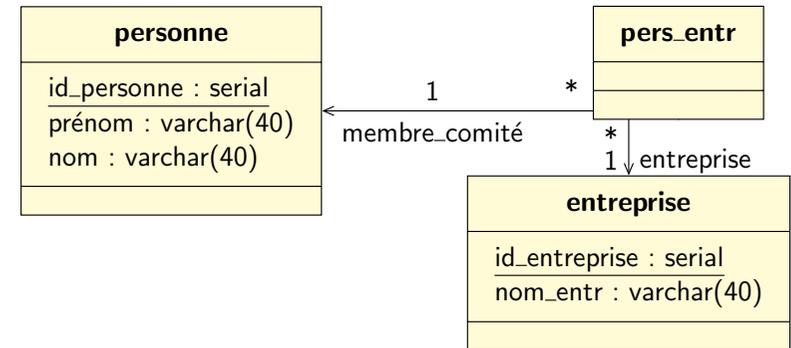


## Cardinalité multiple des deux côtés



- ▶ Chaque lien entre une personne et une entreprise est un couple  $(p, e)$ ; c'est à dire un élément du produit cartésien  $\text{personne} \times \text{entreprise}$ .
- ▶ Donc l'ensemble des liens est une **relation** entre des personne et des entreprise, ou plutôt entre leurs clefs primaires.

## Cardinalité multiple des deux côtés



```

SELECT entreprise FROM pers_entr
WHERE membre_comité=<valeur_id_personne>;

SELECT membre_comité FROM pers_entr
WHERE entreprise=<valeur_id_entreprise>;
    
```

## Petit point technique : les schémas

- ▶ Un **schéma** est un ensemble de table dans une base de données (on peut voir ça comme un dossier).
- ▶ On se trouve par défaut dans un schéma nommé **public**.
- ▶ On les crée en utilisant `CREATE SCHEMA <nom_du_schéma>;`
- ▶ Une table a en réalité pour nom `<nom_du_schéma>.nom_de_la_table`.
- ▶ On peut changer de schéma courant en changeant le `search_path` : `SET search_path TO <nom_du_schéma>;`  
On n'a alors plus besoin de mettre `<nom_du_schéma>.` avant le nom de la table.

## Petit point technique : les schémas

- ▶ Un `CREATE SCHEMA mon_schema;` se de
- ▶ On se `CREATE TABLE mon_schema.ma_table ( public.`
- ▶ On le `column integer PRIMARY KEY);`
- ▶ On le `INSERT INTO mon_schema.ma_table`
- ▶ Une t `<nom VALUES (2), (3), (5), (7);`
- ▶ On peut changer de schéma courant en changeant le `search_path` : `SET search_path TO mon_schema;`  
On n'a alors plus besoin de mettre `<nom_du_schéma>.` avant le nom de la table.

Schémas

M. Sassolas  
L3Pro SCT – M7

Cours 2

Conception

Implémentation

TD/TP

1 Concevoir un schéma de base de donnée

2 Créer les table en SQL

3 TD/TP

Schémas

M. Sassolas  
L3Pro SCT – M7

Cours 2

Conception

Implémentation

TD/TP

↳ C'est l'heure du TD ◀