- **1. Bases de données**
  - » Une base de données est un ensemble d'informations connexes enregistrées dans un dispositif informatique.
  - » Il existe plusieurs types de bases de données (BD) traduisant différentes manières de modéliser l'information
    - BD graphe (NoSQL)
    - BD « array »
    - BD document (NoSQL)
    - BD objet
    - BD relationnelle
    - ...
  - » Tous ces modèles…
    - Sont implémentés avec des outils spécifiques (systèmes de gestion de bases de données)
    - présentent des avantages et des inconvénients
    - couvrent différents domaines d'application

- **2. Le modèle relationnel**
  - **Généralités**
    - » Bien qu'assez ancien (Codd, 1970), le modèle relationnel est encore considéré comme le plus populaire et le plus robuste.
    - » Répond aux objectifs A.C.I.D
      - Atomicité
      - Cohérence
      - Isolation
      - Durabilité
    - » Idéal pour la gestion des transactions bancaires (par exemple)
    - » Peut gérer un très grand nombre de données (plusieurs millions) et d'utilisateurs
    - » Repose sur l'algèbre relationnelle
    - » Il existe de nombreux SGBD pour l'implémentation des BD relationnelles (SGBD relationnels). Par exemple:
      - MySQL (*open source*)
      - PostGreSQL (*open source*)
      - Oracle (*propriétaire*)

– Relation

» Dans une base de données relationnelle les informations sont stockées sous forme de groupes de valeurs : les enregistrements (lignes).

» Un ensemble d'enregistrements relatif à un sujet forme une relation et est stocké dans une table. La base de données comporte une ou plusieurs table(s) et les sujets sont connexes

» Une colonne d'une table = un attribut

➔un enregistrement possède donc un ou plusieurs attributs

» Un attribut est défini par son son nom et son type/domaine (*integer, double, varchar, date, geometry*, …)

» Un attribut peut être soumis à plusieurs contraintes restreignant les opérations sur celui-ci

- Exemples: not null, primary key, …

» Clé primaire = attribut identifiant un enregistrement de façon unique

Livre			
Numéro	**Titre**	**Auteur**	**Catégorie**
452	La géomatique pour les nuls	Laplanche	Informatique
453	Le seigneur des anneaux	Tolkien	Fantaisie
454	L'histoire du rock	Plant	Musique

– Jointure

» Les enregistrements de différentes tables peuvent être associés entre eux grâce au concept de clé étrangère

» Clé étrangère = clé primaire d'une table B présente dans une table A

» Une jointure permet de retrouver les informations d'une table B relative à une table A grâce à sa clé étrangère

» La distribution de l'information en plusieurs tables permet de limiter la redondance

➔Economie d'espace de stockage, meilleures performances de la BD, facilités de mises à jour et maintien de la cohérence des données

Personne		
Matricule	**Nom**	**Prénom**
b734	Donnay	Jean-Paul
e154	Kasprzyk	Jean-Paul

Livre			
Numéro	**Titre**	**Auteur**	**Catégorie**
452	La géomatique pour les nuls	Laplanche	Informatique
453	Le seigneur des anneaux	Tolkien	Fantaisie
454	L'histoire du rock	Plant	Histoire

Emprunt				
Numéro_emprunt	**Matricule_personne**	**Numéro_livre**	**Date_début**	**Date_fin**
1	b734	454	21/02/2011	28/02/2011
2	e154	452	11/09/2010	18/09/2010
3	e154	454	5/10/2010	12/10/2010

- **Langage SQL**
  - » Le langage SQL (« *Structured Query Language* ») permet la manipulation des BD relationnelles
  - » Bien qu'il puisse présenter quelques variantes, SQL est géré par tous les SGBD relationnels
  - » Dans un système d'information, toutes les applications faisant appel à la BD communiquent avec elle au moyen de requêtes SQL
  - » Exemples de fonctionnalités SQL
    - Création de base de données
    - Création de tables
    - Définitions de contraintes (clés primaires, clés étrangères, …)
    - Recherches dans une table sur base d'opérations logiques (« ou », « et »)
    - Insertion dans une table
    - Mise à jour d'un ou plusieurs enregistrement(s)
    - Agrégations
    - …

- **3. Quelques requêtes SQL**
  - **Création d'une table avec clé primaire**

```
CREATE TABLE personne
(
        matricule character varying NOT NULL,
        nom character varying,
        prenom character varying,
        CONSTRAINT "Personne_pkey" PRIMARY KEY (matricule)
)
```

	matricule character varying	nom character varying	prenom character varying

» NB: la contrainte de clé primaire empêche l'existence de deux enregistrements (personnes) avec le même matricule

– Insertion d'un enregistrement dans une table

INSERT INTO personne(matricule, prenom, nom)
VALUES ('a213', 'Roland', 'Billen');
INSERT INTO personne(matricule, prenom, nom)
VALUES ('b734', 'Jean-Paul', 'Donnay');
INSERT INTO personne(matricule, prenom, nom)
VALUES ('e154', 'Jean-Paul', 'Kasprzyk');

	matricule [PK] character varying	nom character varying	prenom character varying
1	a213	Billen	Roland
2	b734	Donnay	Jean-Paul
3	e154	Kasprzyk	Jean-Paul

– Recherche dans une table

```
SELECT matricule
FROM personne
WHERE prenom='Jean-Paul' AND nom ='Kasprzyk'
```

	matricule character varying
1	e154

```
SELECT *
FROM personne
WHERE nom ='Billen' or nom ='Donnay'
```

	matricule character varying	nom character varying	prenom character varying
1	b734	Donnay	Jean-Paul
2	a213	Billen	Roland

– Création d'une table avec clés étrangères

» Création de la table « emprunt » associée à « personne » et « livre »

```
CREATE TABLE public.emprunt(
    numero_emprunt integer NOT NULL,
    matricule_personne character varying,
    numero_livre integer,
    date_debut date,
    date_fin date,
    CONSTRAINT emprunt_pkey PRIMARY KEY (numero_emprunt),
    CONSTRAINT emprunt_matricule_personne_fkey FOREIGN KEY (matricule_personne)
    REFERENCES personne (matricule) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION,
    CONSTRAINT emprunt_numero_livre_fkey FOREIGN KEY (numero_livre)
    REFERENCES livre (numero) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION
    )
```

	numero_emprunt integer	matricule_personne character varying	numero_livre integer	date_debut date	date_fin date

» NB: la contrainte de clé étrangère empêche la création d'un emprunt dont l'emprunteur ou le livre sont absent de la BD

– Jointure

» Recherche de tous les livres empruntés par Kasprzyk

```
SELECT livre.titre
FROM livre, personne, emprunt
WHERE nom ='Kasprzyk'
AND emprunt.matricule_personne=personne.matricule
AND livre.numero=emprunt.numero_livre
```

	titre character varying
1	La géomatique pour les nuls
2	L'histoire du rock

– Jointure: autre syntaxe équivalente

» Recherche de tous les livres empruntés par Kasprzyk

```
SELECT livre.titre
FROM livre
INNER JOIN emprunt
ON livre.numero=emprunt.numero_livre
INNER JOIN personne
ON personne.matricule=emprunt.matricule_personne
WHERE personne.nom='Kasprzyk'
```

	titre character varying
1	La géomatique pour les nuls
2	L'histoire du rock

– Jointure: « inner join » et « left outer join »

» Jointure entre « livre » et « emprunt »

```
SELECT *
FROM livre
INNER JOIN emprunt
ON livre.numero=emprunt.numero_livre
```

Liste uniquement les enregistrements joints

numero integer	titre character varying	auteur character vary	categorie character vary	numero_emprunt integer	matricule_personne character varying	numero_li integer	date_debut date	date_fin date
454	L'histoire du Rock	Plant	Histoire	1	b734	454	2011-02-21	2011-02-28
452	La géomatique pour le...	Laplanche	Informatique	2	e154	452	2010-09-11	2010-09-18
454	L'histoire du Rock	Plant	Histoire	3	e154	454	2010-10-05	2010-10-12

```
SELECT *
FROM livre
LEFT OUTER JOIN emprunt
ON livre.numero=emprunt.numero_livre
```

Liste aussi les enregistrements non-joints de la table de gauche (« livre »)

numero integer	titre character varying	auteur character vary	categorie character varyi	nume intege	matricule character	numero integer	date_debut date	date_fin date
454	L'histoire du Rock	Plant	Histoire	1	b734	454	2011-02-21	2011-02-28
452	La géomatique pour le...	Laplanche	Informatique	2	e154	452	2010-09-11	2010-09-18
454	L'histoire du Rock	Plant	Histoire	3	e154	454	2010-10-05	2010-10-12
453	Le Seigneur des Annea...	Tolkien	Fantaisie	[null]	[null]	[null]	[null]	[null]

– Jointure: « right outer join »

» Jointure entre « livre » et « emprunt »

```
SELECT *
FROM livre
RIGHT OUTER JOIN emprunt
ON livre.numero=emprunt.numero_livre
```

Liste aussi les enregistrements non-joints de la table de droite (« emprunt »)
➔ Inutile dans ce cas-ci puisque la clé étrangère pointant de la table « livre » vers la table « emprunt » ne permet pas des emprunts sans livre présents dans la table « livre »

numero integer	titre character varying	auteur character vary	categorie character vary	numero_emprunt integer	matricule_personne character varying	numero_li integer	date_debut date	date_fin date
454	L'histoire du Rock	Plant	Histoire	1	b734	454	2011-02-21	2011-02-28
452	La géomatique pour le...	Laplanche	Informatique	2	e154	452	2010-09-11	2010-09-18
454	L'histoire du Rock	Plant	Histoire	3	e154	454	2010-10-05	2010-10-12

– Jointure et doublons

» Recherche de tous les titres de livres ayant été empruntés

```
SELECT livre.titre
FROM livre, emprunt
WHERE
livre.numero=emprunt.numero_livre
```

titre character varying
L'histoire du Rock
La géomatique pour le...
L'histoire du Rock

Puisque « l'histoire du rock » a été emprunté deux fois, il apparaît deux fois dans la liste

```
SELECT DISTINCT livre.titre
FROM livre, emprunt
WHERE
livre.numero=emprunt.numero_livre
```

Suppression des doublons dans la requête

titre character varying
L'histoire du Rock
La géomatique pour le...

– Mise à jour

UPDATE livre
SET titre = 'La géomatique pour les pros'
WHERE numero=452

	numero [PK] integer	titre character varying	auteur character varying	categorie character varying
1	452	La géomatique pour les pros	Laplanche	Informatique
2	453	Le seigneur des anneaux	Tolkien	Fantasie
3	454	L'histoire du rock	Plant	Musique

– Suppression d'un enregistrement

```
DELETE
FROM personne
WHERE matricule='b734'
```

Opération impossible en raison de la clé étrangère de la table emprunt

On ne peut supprimer un enregistrement d'une table sans supprimer tous ses enregistrements associés dans d'autres tables via une clé étrangère
➔ cohérence de la BD

```
ERREUR:  UPDATE ou DELETE sur la table « personne » viole la contrainte de clé étrangère
« emprunt_matricule_personne_fkey » de la table « emprunt »
DETAIL:  La clé (matricule)=(b734) est toujours référencée à partir de la table « emprunt ».
********** Error **********
```

– Agrégation

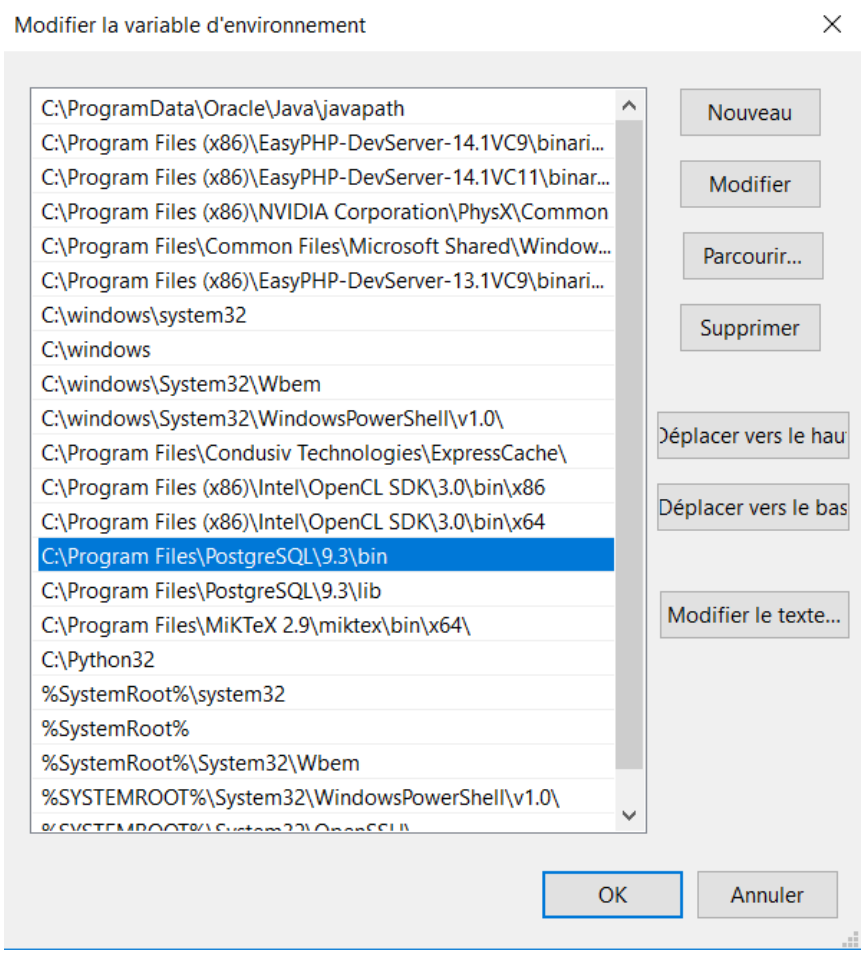
» Comptage du nombre d'emprunts par personnes

```
SELECT matricule_personne, count(numero_emprunt)
FROM emprunt
GROUP BY matricule_personne
```

	matricule_personne character varying	count bigint
1	b734	1
2	e154	2

- **4. SGBD PostGreSQL (Postgres)**
  - **Généralités**
    - » Système de Gestion de Base de Données relationnel objet
      - Données stockées sous formes de relations manipulables en SQL
      - Intégration de types (classes) permettant le stockage et la manipulations d'objets complexes au sein de la structure relationnelle (SQL3) → cfr chapitre 3
      - Données facilement exploitables dans un langage de programmation orienté objet grâce à l'utilisation du pilote approprié (ex: Psycopg2 pour Python, JDBC pour Java, …)
    - » Système *open source* fondé sur une communauté mondiale de développeurs
    - » Evolue depuis plus de 30 ans avec nouvelles versions régulières (version actuelle: 11)
    - » Permet la création de fonctions complexes grâce au langage procédural PL/pgSQL
      - Contrairement à SQL, PL/pgSQL permet l'utilisation de boucles et de conditions sur des variables
    - » Idéal pour la gestion de données spatiales grâce à son extension PostGIS

– **Interface PSQL**
  - » Interface basique en ligne de commandes
  - » ex: invite de commandes Windows avec association préalable du répertoire « bin » de postgres à la variable d'environnement « path »
  - » Peut être appelée par un programme externe



» Connexion à une base de données (distante ou locale)

Utilisateur
Host
Port
Nom de la base de données

```
D:\>psql -U postgres -h localhost -p 5432 -d test
Mot de passe pour l'utilisateur postgres :
```

- La connexion à une base de données nécessite toujours au minimum ces 4 paramètres (valable pour tous les SGBD)
- L'adresse (host) peut être l'adresse IP du serveur ou son nom de domaine
- Le port logiciel est le port de communication du serveur pour envoyer et recevoir des informations
  - Par défaut, postgres communique avec le port 5432
  - N.B: Un site internet communique généralement à travers le port 8000

» Exécution d'une requête SQL

```
test=# select * from personne;
 pk_personne |    nom     |   prenom   | sexe | date_naissance
-------------+------------+------------+------+----------------
 85022126136 | Beaumet    | Julien     | M    | 1970-11-11
 85022126137 | Bodeux     | Sarah      | F    | 1971-11-11
 85022126138 | Bourlard   | Emilie     | F    | 1972-11-11
 85022126139 | Delmot     | Justine    | F    | 1973-11-11
 85022126140 | Dethier    | Perrine    | F    | 1974-11-11
```

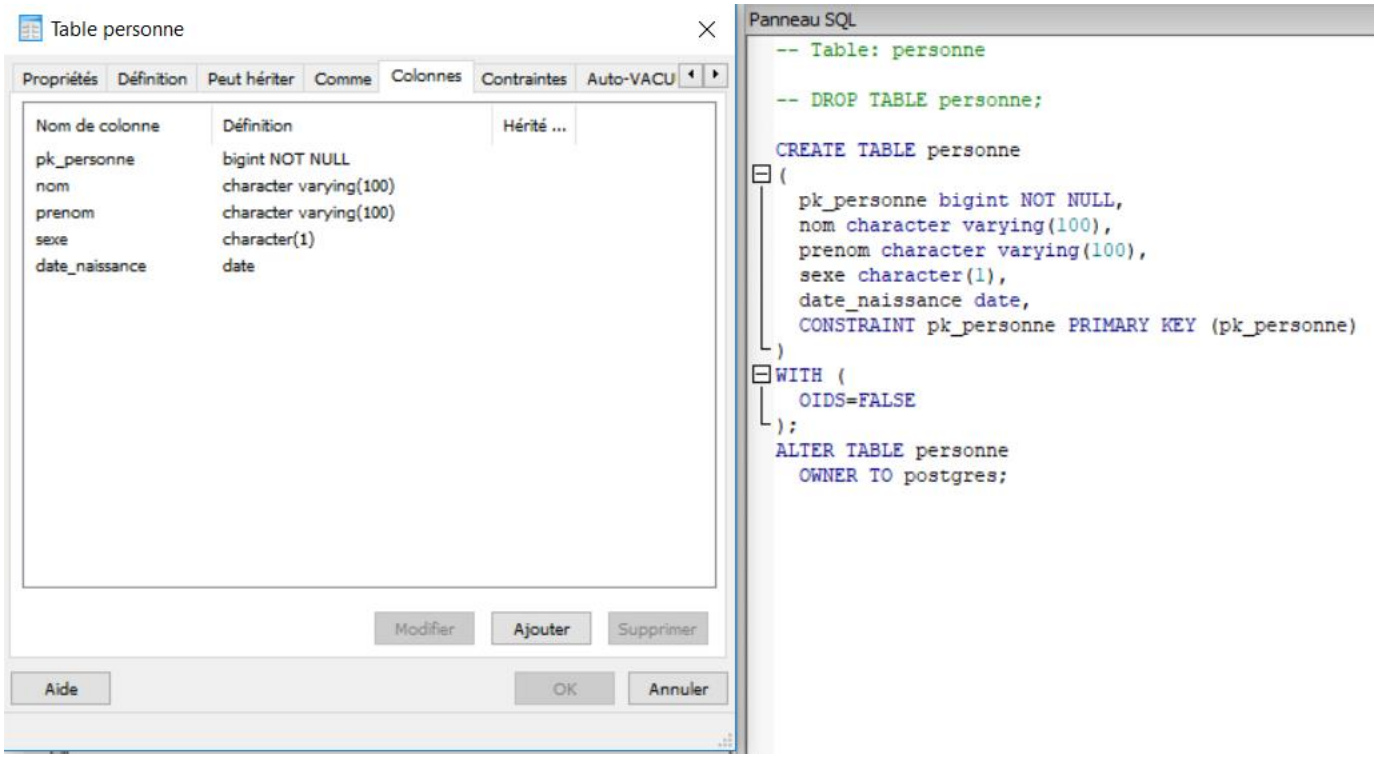
- Note: psql permet aussi l'exécution d'un fichier sql contenant une série d'instructions

– Interface PGAdmin

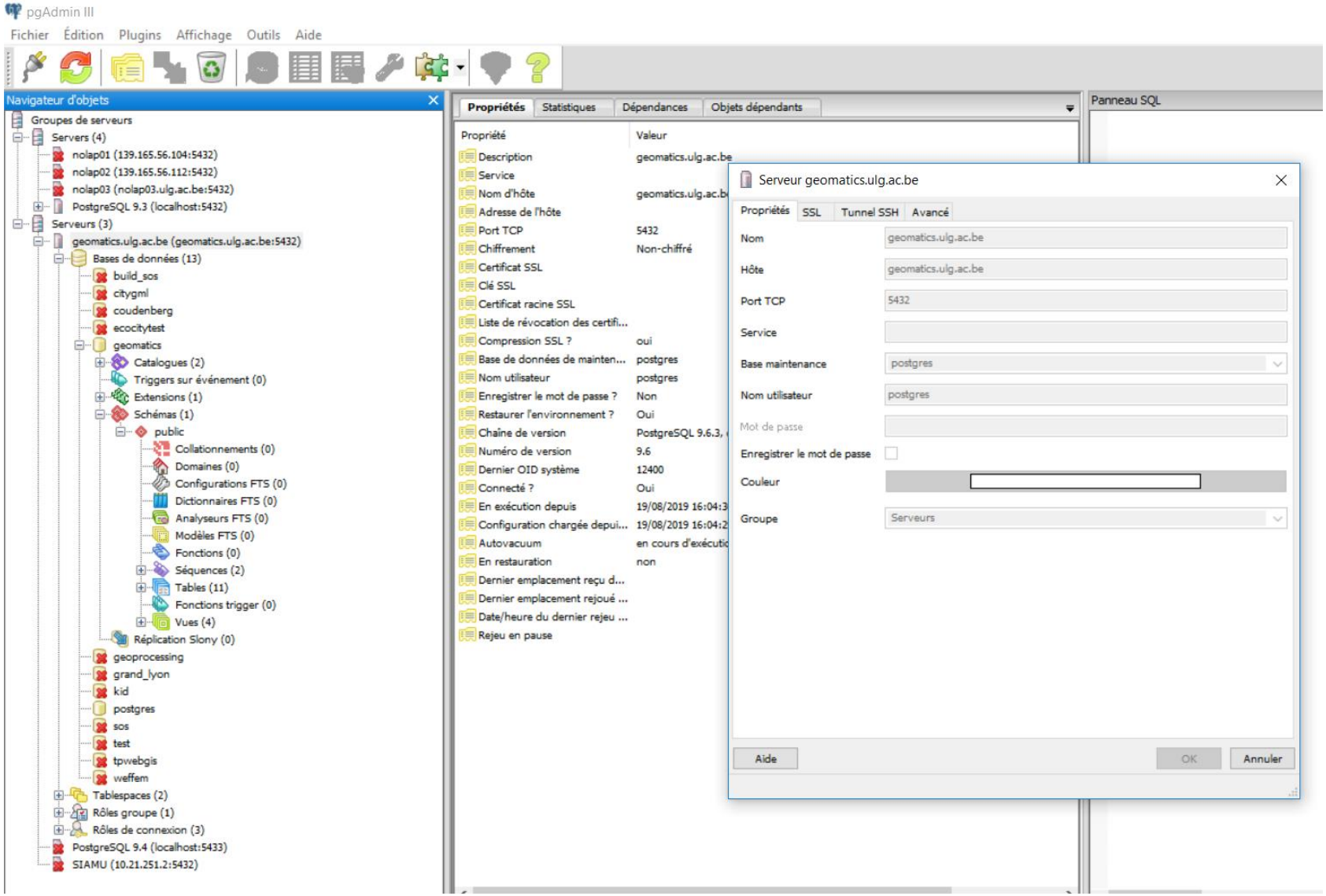
» Outil complet et convivial pour l'administration de BD postgres

» Permet la manipulation des différents éléments d'une base de données de manière graphique

» Derrière chaque opération dans l'interface se cache une requête SQL

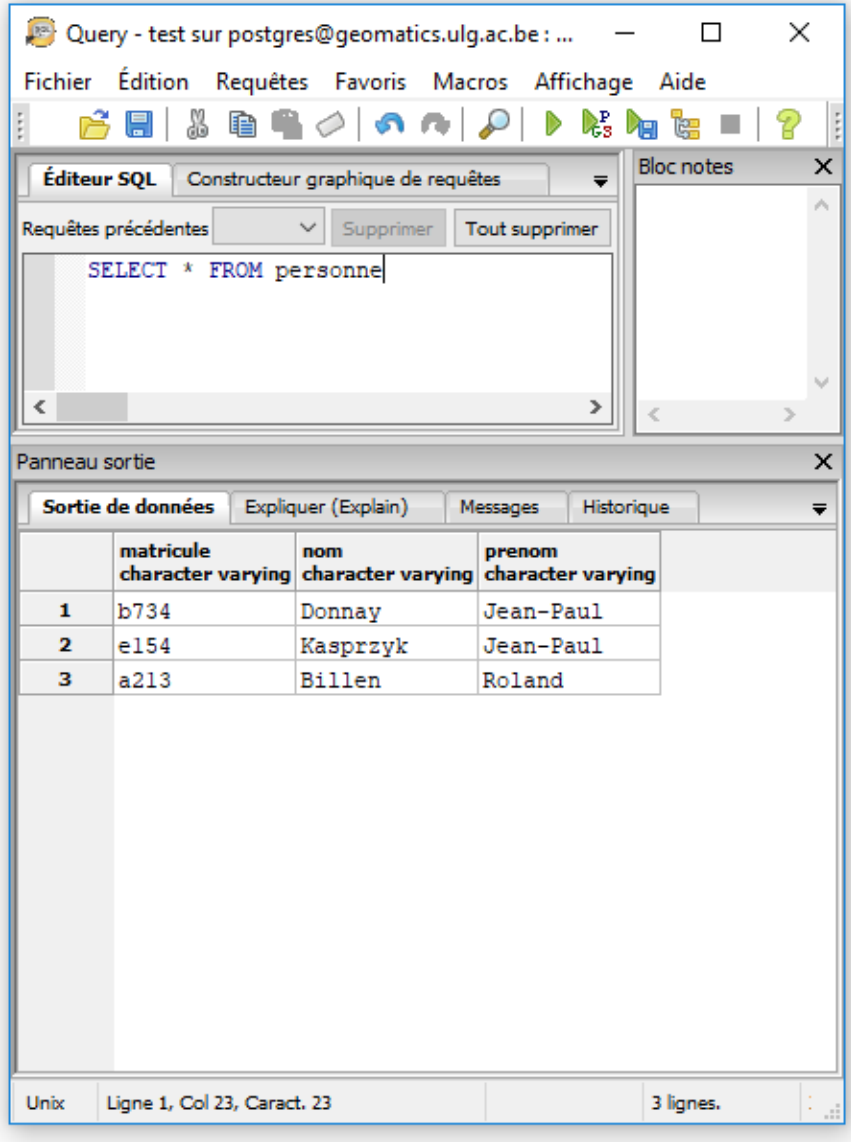


Création d'une table via PGAdmin III et requête SQL associée



Connexion à une BD distante à travers PGAdmin III

» En plus de ses nombreux modules d'administration (« presse bouton »), PGAdmin dispose de sa propre console SQL



– **Rôle**

» La connexion à une base de données nécessite toujours de passer par un rôle

» Un rôle correspond à un utilisateur ou un groupe d'utilisateurs ayant des droits (ou privilèges) particuliers sur une BD

» Un rôle peut être propriétaire d'une BD ou d'un élément de la BD (ex: table) et attribuer des droits à d'autres rôles sur ces éléments. Par exemple:

- Droit en lecture (SELECT)
- Droit de mise à jour des données (UPDATE)
- Droit d'ajouter des données (INSERT)
- Droit de supprimer des données (DELETE / TRUNCATE)

» L'attribution de rôles est indispensable pour assurer la sécurité et la cohérence de la BD lorsqu'elle implique différents utilisateurs

» Un serveur postgres implique toujours un « super utilisateur » ayant tous les droits sur toutes les BD. Celui-ci se nomme simplement « postgres ».

– Schéma

- » Un serveur postgres peut contenir plusieurs bases de données mais celles-ci ne peuvent pas directement interagir entre elles
- » Les schémas permettent de regrouper différentes tables ensemble tout en permettant des interactions entre différents schémas (exemple: jointure)
- » Plus facile de s'y retrouver lorsque, par exemple, le SI comprend plusieurs applications (une application par schéma)
- » Pas de conflits de noms de tables entre différents schémas
- » Le schéma par défaut de postgres s'appelle « public »
- » A l'exception du schéma « public », l'appel d'une table en SQL nécessite la spécification du nom de son schéma. Par exemple:

```
SELECT * FROM «2015».personne
```

– **Vue**

» Une vue est une table « dématérialisée » permettant de mémoriser le résultat d'une requête. Par exemple:

```
CREATE or REPLACE VIEW femme AS
        SELECT *
        FROM personne
        WHERE sexe='f'
```

» L'appel de la vue « femme » ne montrera que les informations relatives aux femmes de la table « personne »

» La vue ne consomme aucun espace mémoire, seul la requête est mémorisée

» Une vue n'est accessible qu'en lecture (pas de modification des données)

– Fonction

» Postgres permet la création de fonctions directement utilisables dans des requêtes SQL

» Ces fonctions peuvent être écrites en plusieurs langages, notamment SQL et PL/pgSQL

» Exemple: création d'une fonction « add » pour additionner deux valeurs d'attributs au sein d'une requête SQL

```
CREATE FUNCTION add(integer, integer) RETURNS integer
        AS 'select $1 + $2;'
        LANGUAGE SQL
        IMMUTABLE
        RETURNS NULL ON NULL INPUT;
```

» Exemple d'utilisation de la fonction « add »

```
SELECT add(nb_emploi_femme, nb_emploi_homme) as nb_emploi_total
FROM entreprise
```

- **Fonction trigger**
  - » Une fonction trigger est une fonction appelée automatiquement lors d'un événement
    - Par exemple: alimentation d'un historique en cas de mise à jour d'une donnée
- **Séquence**
  - » Une séquence permet l'incrémentation automatique d'un attribut lorsqu'une nouvelle données est insérée
    - Par exemple: incrémentation automatique d'un nombre entier en guise d'identifiant (clé primaire)
- **Extension**
  - » Il existe un certain nombre d'extensions installables avec postgres. Par exemple
    - PostPic pour le traitement d'images
    - PostGIS pour la gestion des données spatiales
    - PL/pgSQL installé par défaut

- **5. Exercice: création d'une base de données postgres via PGADMIN**

Personne		
Matricule	**Nom**	**Prénom**
b734	Donnay	Jean-Paul
e154	Kasprzyk	Jean-Paul

Livre			
Numéro	**Titre**	**Auteur**	**Catégorie**
452	La géomatique pour les nuls	Laplanche	Informatique
453	Le seigneur des anneaux	Tolkien	Fantaisie
454	L'histoire du rock	Plant	Histoire

Emprunt				
Numéro_emprunt	**Matricule_personne**	**Numéro_livre**	**Date_début**	**Date_fin**
1	b734	454	21/02/2011	28/02/2011
2	e154	452	11/09/2010	18/09/2010
3	e154	454	5/10/2010	12/10/2010

- **6. Modélisation d'une base de données**
  - **Généralités**
    - » La conception d'une base de données implique d'abord l'élaboration de modèles (ou schémas) pour:
      - Offrir une vision synoptique de l'ensemble de la BD
      - Faciliter la communication et la compréhension entre les différents acteurs de conception du SI(G): concepteurs, développeurs, futurs utilisateurs, …
      - Faciliter la maintenance de la BD
    - » Il existe 3 grands types de modèles allant du niveau le plus abstrait (conceptuel) au niveau le plus concret (physique). Dans l'ordre de conception:
      - 1. Modèle conceptuel de données ➔ système
      - 2. Modèle logique de données ➔ base de données
      - 3. Modèle physique de données ➔ machine
    - » Outils CASE (Computer-Aided Software Engineering)
      - Outils permettant la conversion automatique de modèles conceptuels en modèles logiques et/ou physiques
      - Exemples: ArgoUML, Eclipse, Moskitt

– Modèle conceptuel de données (MCD)

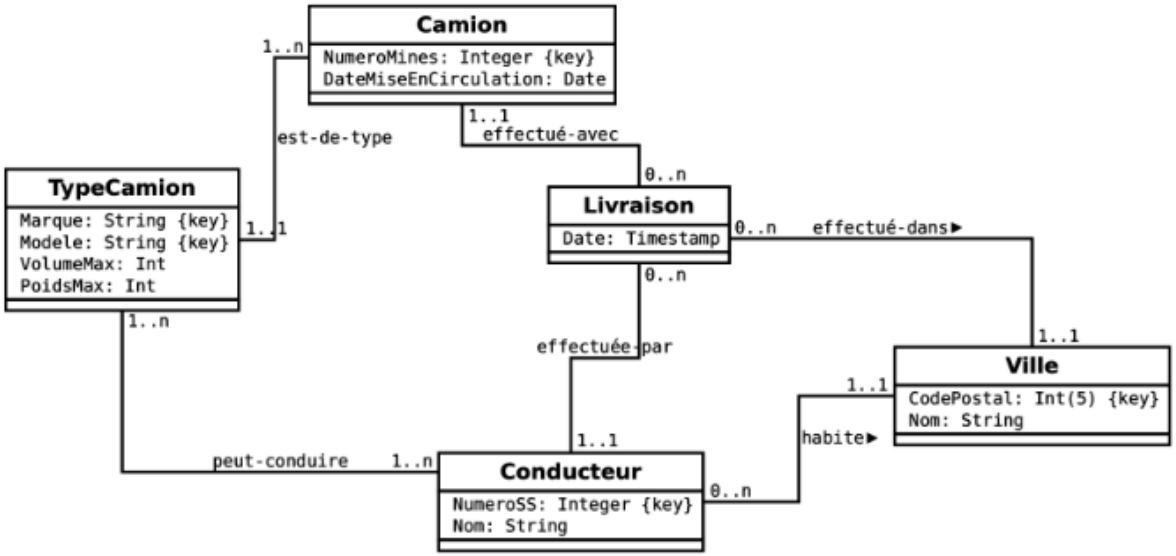
» Description des différentes entités (ou classes) du système et de leurs associations (≠ relations)

» Exploite généralement un formalisme graphique

- Exemple: modèle entité-association, diagramme de classes UML

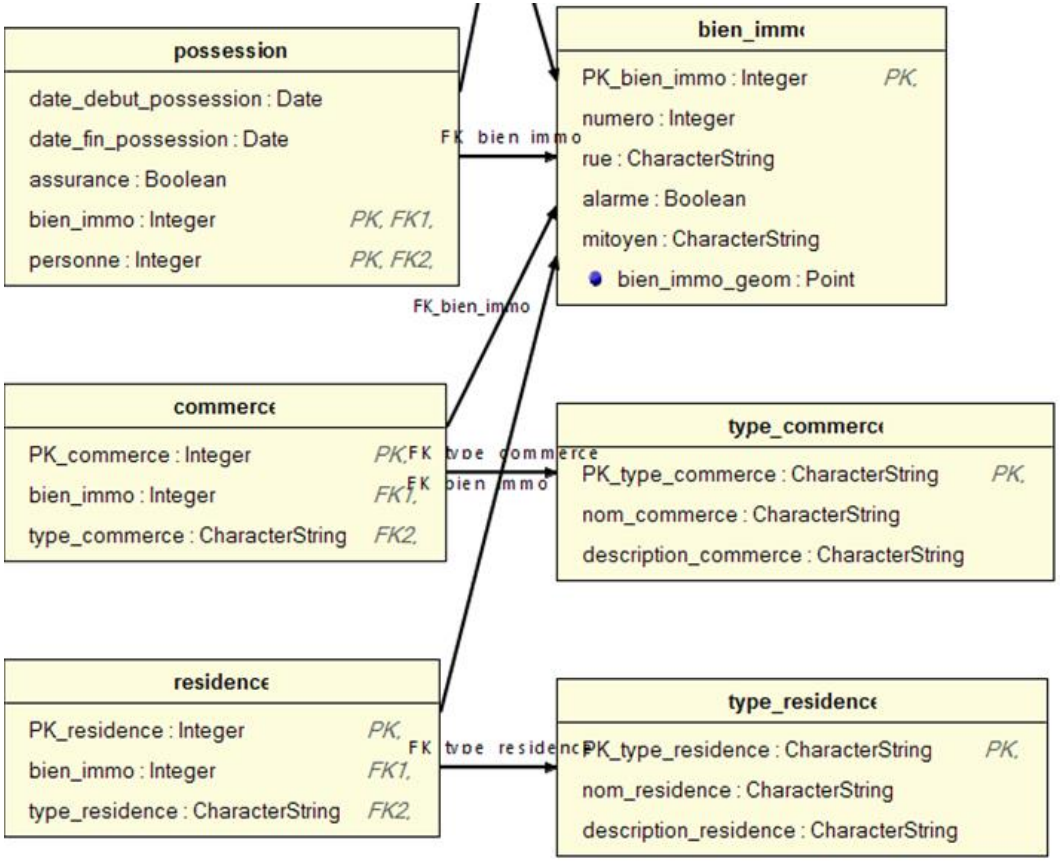
» Pas de notion de clé étrangère → on ne parle pas encore de BD relationnelle

» Indépendant des outils utilisés pour l'implémentation du SI ou de la BD



Exemple de MCD: diagramme de classes UML (Source: Crozat, 2018)

– Modèle logique de données (MLD)



Exemple de MLD

- » Description des données découlant du MCD sans faire référence à un langage de programmation
- » Pour la modélisation d'une BD relationnelle, on ne parle plus d'entités mais de relations (ou tables)
- » Description des clés primaires et étrangères
- » Dépendant de la technologie de stockage des données (ex: base de donnes relationnelle)
- » Peut exploiter un formalisme graphique ou non

– Modèle physique de données (MPD)

» Description du système dans un langage de programmation interprétable par la machine

» Pour la modélisation d'une BD relationnelle, le MPD est décrit en SQL

- Description des tables, attributs, contraintes, …

» Dépendant du SGBD utilisé

```
CREATE TABLE autoroutes
(
  nom_b text,
  nom_e text,
  autocode text NOT NULL,
  CONSTRAINT autoroutes_pkey PRIMARY KEY (autocode)
);
CREATE TABLE troncons
(
  longueur_troncon real,
  id_troncon integer NOT NULL,
  debute_id_noeud integer NOT NULL,
  termine_id_noeud integer NOT NULL,
  autocode text,
  CONSTRAINT troncons_pkey PRIMARY KEY (id_troncon),
  CONSTRAINT autoroute FOREIGN KEY (autocode)
     REFERENCES autoroutes (autocode) MATCH SIMPLE,
);
```

Exemple de MPD

- **7. Diagramme de classes UML**
  - **UML**
    - » *Unified Modeling Langage*
    - » Formalisme très répandu pour la modélisation graphique des vues statiques et dynamiques d'un système
    - » Utilisé pour le développement logiciel et en conception orientée objet
    - » UML propose 14 types de diagrammes pour la modélisation d'un projet tout au long de son cycle de vie, notamment:
      - Diagrammes statiques
        - **Diagrammes de classes**: représentation des classes intervenant dans le système (statique)
          → Utilisé, entre autres, pour la modélisation conceptuelle d'une BD relationnelle (MCD)
        - Diagrammes de cas d'utilisation: interaction entre système et acteurs
      - Diagrammes dynamiques
        - Diagrammes de séquences: modélisation des traitements

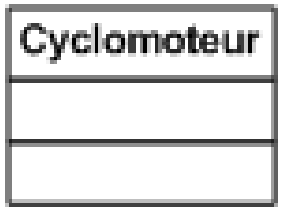
– Classe

» La classe représente la description d'un ensemble d'objets possédant les mêmes caractéristiques.

» Un objet est une entité aux frontières bien définies, possédant une identité et encapsulant un état et un comportement. Un objet est une « instance » ou occurrence d'une classe.

- Ex : Jean-Paul Kasprzyk est une instance de la classe PersonnelULg.
- Voiture, patient, commune, état, peuvent être des classes

» Représentée par une boite divisée en 3 parties



– Attributs et opérations

» Un attribut représente un type d'information contenu dans une classe.

- Ex : vitesse courante, cylindrée, numéro de parcelle, surface,…

» Une méthode représente un élément de comportement dynamique contenu dans une classe. Les méthodes ne sont pas exploitées dans les bases de données relationnelles mais uniquement en programmation orientée objet

» Les attributs et les méthodes sont représentés dans les 2 cases restantes de la classe.

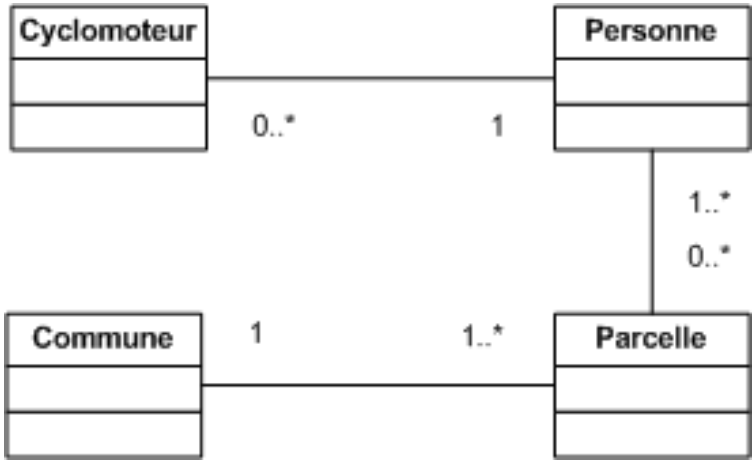
– Association

» Une association représente un lien sémantique durable et bidirectionnel entre deux classes.

- Ex : Une personne peut posséder des voitures.
  « Possède » (ou « est possédé par ») est une association entre les classes Personne et Parcelle.

» Aux extrémités d'une association, on fait figurer une indication de multiplicité (cardinalité). Elle spécifie sous la forme d'un intervalle d'entiers positifs ou nuls le nombre d'objets qui peuvent participer à une relation avec un objet de l'autre classe dans le cadre d'une association.

- Ex : Une personne peut posséder plusieurs parcelles ; une parcelle est possédée par une seule personne (quoique…)
- En BD relationnelle, les seules cardinalités modélisables sont:
  - 0..*
  - 1..*
  - 0..1
  - 1..1
- « * », également noté « N », signifie « un entier strictement supérieur à 1 »

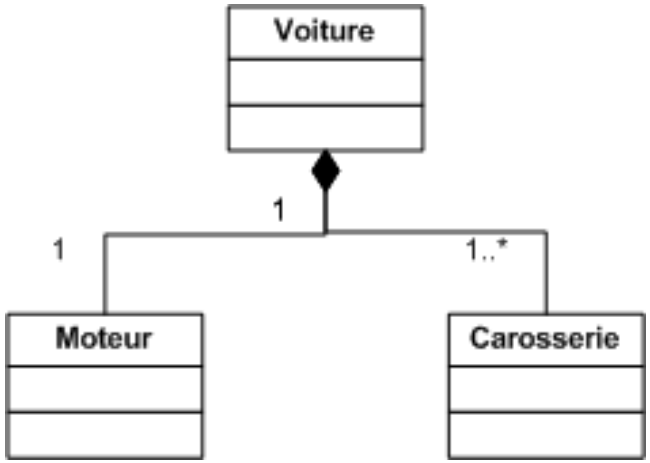


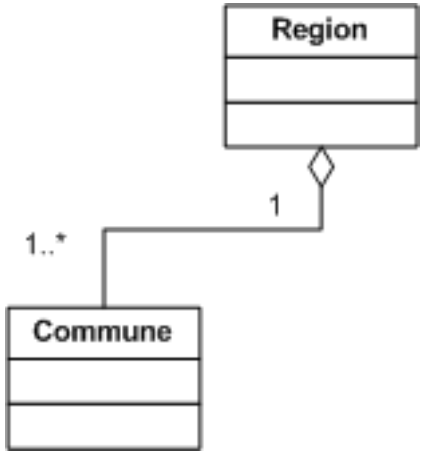
– Agrégation et composition

» Une agrégation est un cas particulier d'association non symétrique exprimant une relation de contenance. Les agrégations n'ont pas besoin d'être nommées : implicitement elles signifient « contient », « est composé de ».

» Une composition est une agrégation plus forte impliquant que :

- Un élément ne peut appartenir qu'à un seul agrégat composite (agrégation non partagée);
- La destruction de l'agrégat composite entraine la destruction de tous ses éléments (le composite est responsable du cycle de vie des parties).

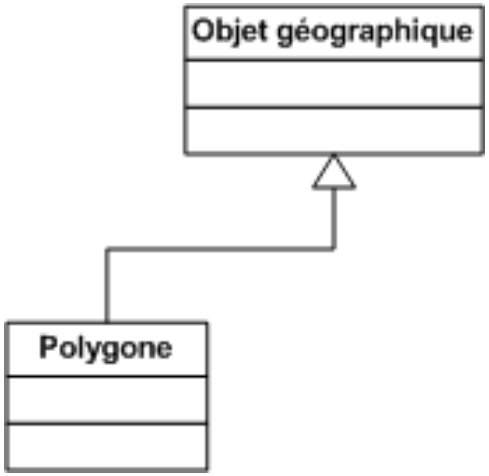


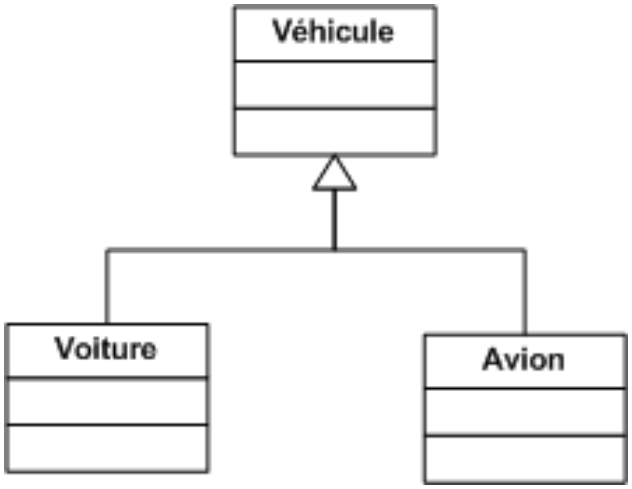


– Généralisation – spécialisation

» Une super-classe est une classe plus générale reliée à une ou plusieurs autres classes plus spécialisées (sous-classes) par une relation de généralisation. Les sous-classes « héritent » des propriétés de leur super-classe et peuvent comporter des propriétés spécifiques supplémentaires.

- Ex : les voitures, les bateaux et les avions sont des moyens de transport. Ils possèdent tous une marque, un modèle,… par contre, seuls les bateaux ont un tirant d'eau,…

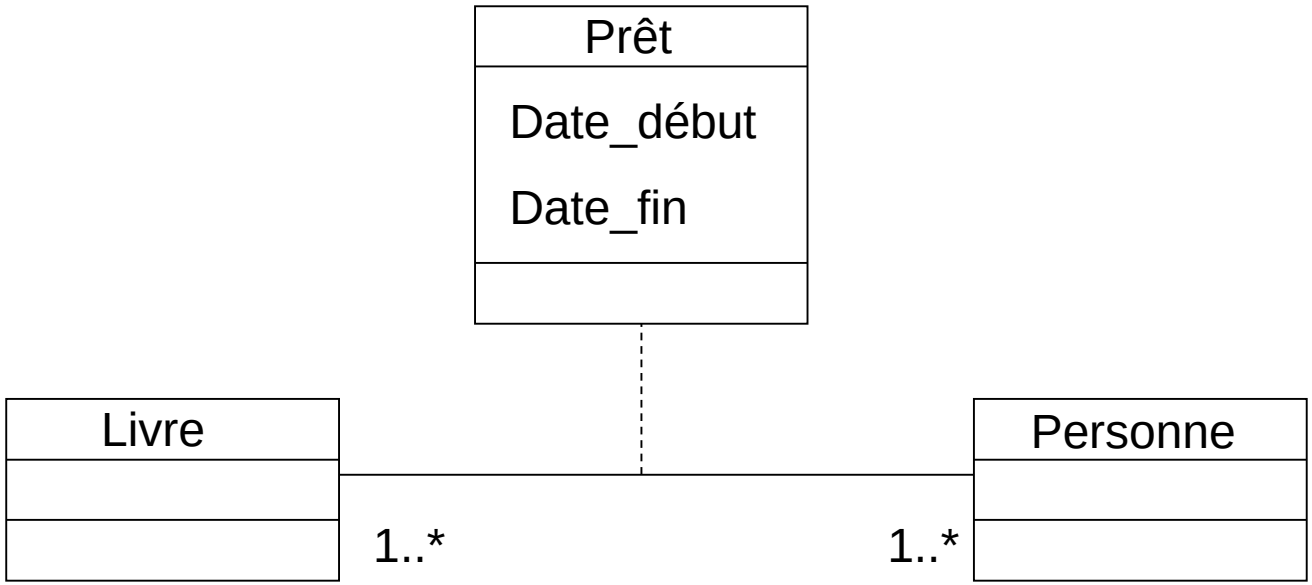




– **Classe-association**

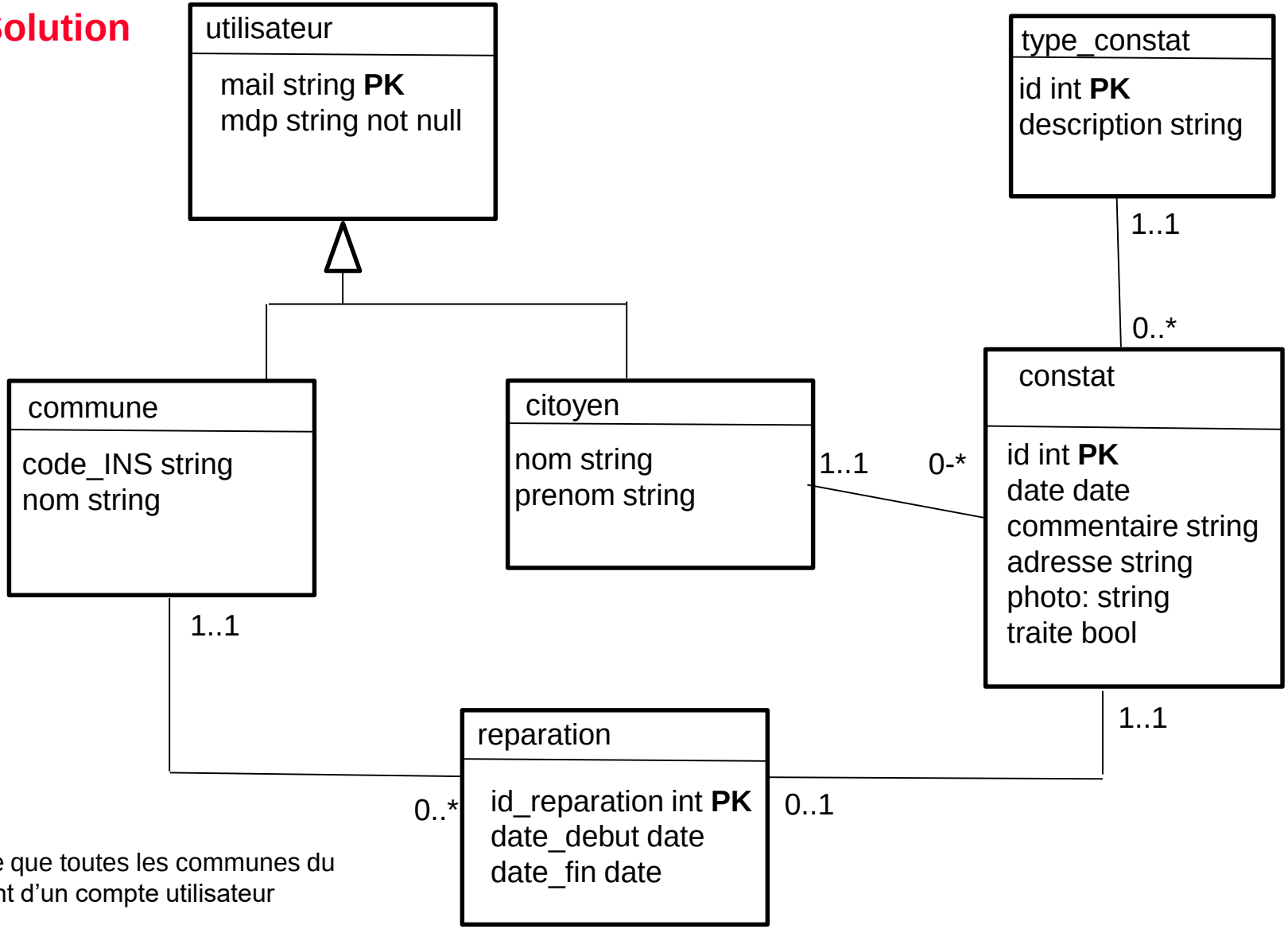
» Une classe-association est une association pouvant, comme une classe, contenir des attributs

- Exemple: une association « prêt » entre une classe « livre » et une classe « personne » peut contenir les attributs « date_début » et « date_fin »



- **8. Exercice: modélisation UML d'une base de données relationnelle**
  - **Enoncé**
    - » Il vous est demandé de concevoir un diagramme de classes UML pour la modélisation de la base de données d'une application type « BetterStreet »
    - » L'application permet à des citoyens de signaler des problèmes sur le terrain (ex: nid de poule) nécessitant une intervention de la commune concernée
    - » L'application gère des utilisateurs pouvant s'identifier avec leur adresse e-mail et un mot de passe. Un utilisateur peut être soit un citoyen, caractérisé par un nom et un prénom, soit une commune, caractérisée par un code INS et un nom (ex: Liège)
    - » Un citoyen peut signaler un constat dont le type serait choisi dans une liste prédéfinie (ex: nid de poule, arbre tombé, etc…). Un constat serait également caractérisé par une date, une photo (fichier jpg associé à une adresse de répertoire), une adresse et un commentaire éventuel.
    - » Une commune peut effectuer une réparation sur un constat signalé sur son territoire. Une réparation est caractérisée par une date de début et une date de fin. Après la réparation, le constat peut être signalé comme traité par la commune.

– Solution



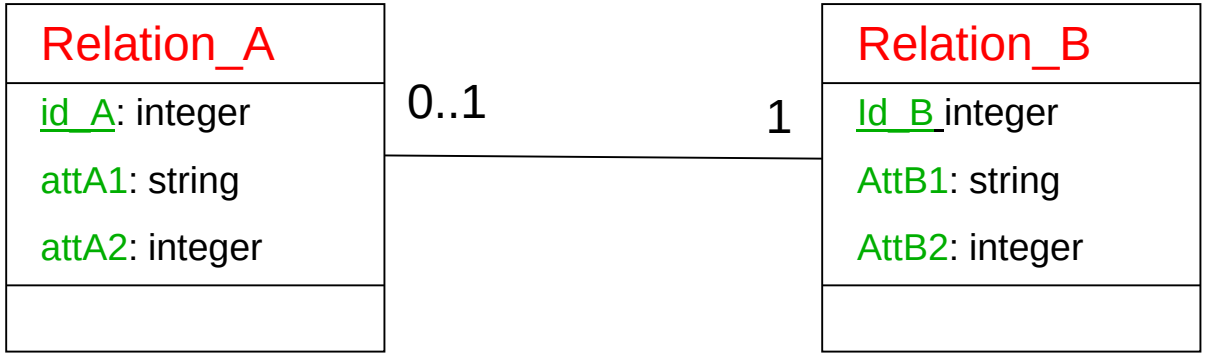
Note: on suppose que toutes les communes du territoire disposent d'un compte utilisateur

- **9. De UML à SQL**
  - **Classe avec attributs**

Relation_A
num_relation_a: integer Att1: string Att2: integer

```
create table relation_A (
        num_relation_a integer primary key,
        att1 text,
        att2 integer ) ;
```

– Association 1 à 1 (dépendance fonctionnelle)

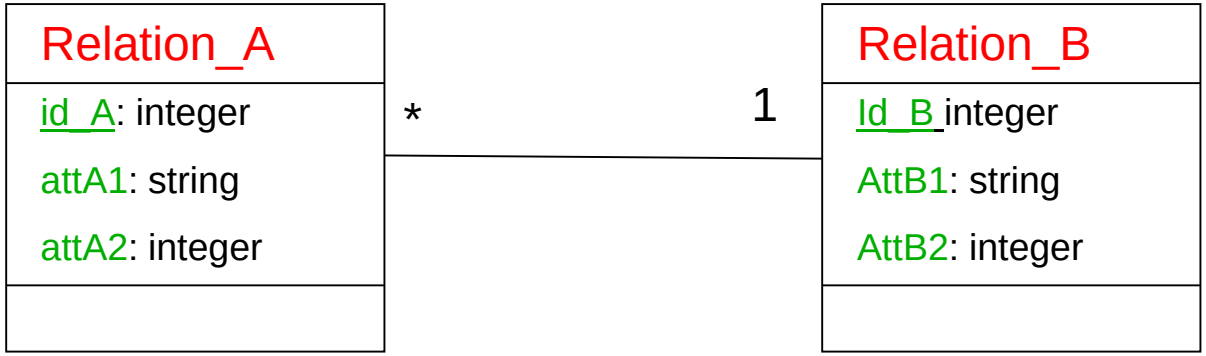



```
create table relation_B (
        id_B integer primary key,
        attB1 text,
        attB2 integer );
create table relation_A (
        id_A integer primary key,
        num_B integer references relation_B,
        attA1 text,
        attA2 integer );
```

La clé étrangère se place dans une des deux tables au choix

En cas d'une présence de cardinalité 0..1, il est préférable de placer la clé étrangère de ce côté afin d'éviter la multiplication de valeurs « NULL »

– Association 1 à N




```
create table relation_B (
        id_B integer primary key,
        attB1 text,
        attB2 integer );
create table relation_A (
        id_A integer primary key,
        num_B integer references relation_B,
        attA1 text,
        attA2 integer );
```

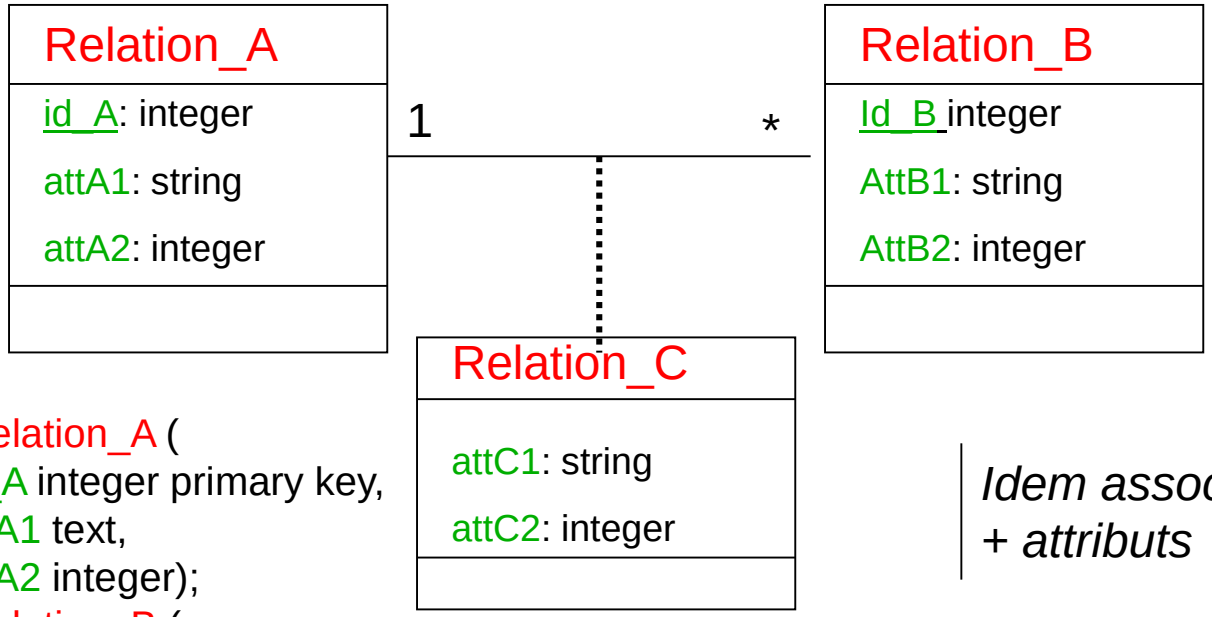
Association 1 à N: la clé étrangère se place du côté du N

– Association N à N


```
create table relation_A (
        id_A integer primary key,
        attA1 text,
        attA2 integer);
create table relation_B (
        id_B integer primary key,
        attB1 text,
        attB2 geometry);
Create table relation_A_B (
        num_A integer references relation_A,
        num_B integer references relation_B,
        primary key (num_A, num_B));
```

Création d'une table supplémentaire contenant les deux clés étrangères

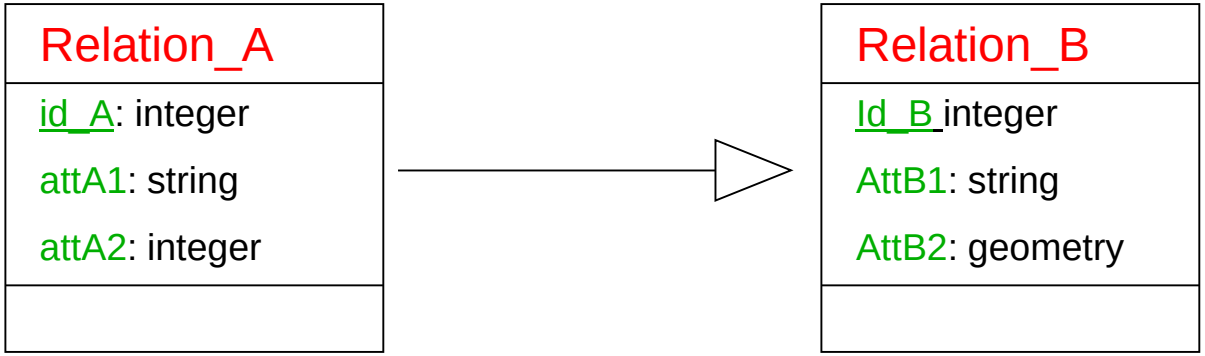
– Classe-association



Idem association N à N + attributs

```
create table relation_A (
        id_A integer primary key,
        attA1 text,
        attA2 integer);
create table relation_B (
        id_B integer primary key,
        attB1 text,
        attB2 geometry);
Create table relation_C (
        num_A integer references relation_A,
        num_B integer references relation_B,
        attC1 text,
        attC2 integer,
        primary key (num_A, num_B));
```

– Généralisation




```
create table relation_B (
        id_B integer primary key,
        attB1 text,
        attB2 integer );
create table relation_A (
        id_A integer primary key,
        num_B integer references relation_B,
        attA1 text,
        attA2 integer );
```

La clé étrangère se trouve dans la classe « fille »

Note: ceci est une des manières de modéliser la généralisation en SQL. Il en existe d'autres…