جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL

ENSAM
المدرسة الوطنية العليا للفنون و المهن
ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS

JOBINTECH

Bases de données

Ahmed Laatabi
a•laatabi{at}umi•ac•ma
ENSAM - Meknès
2025-2026

Problème

Des étudiants assistent à des cours dispensés par des professeurs dans le cadre de formations organisées dans les locaux de l'université Moulay Ismaïl.

Les cours peuvent appartenir aux domaines de l'économie, des langues ou de l'informatique, et ont une durée définie en heures.

Chaque cours possède des objectifs spécifiques à atteindre.

Comment peut-on stocker les données relatives à cette situation ?

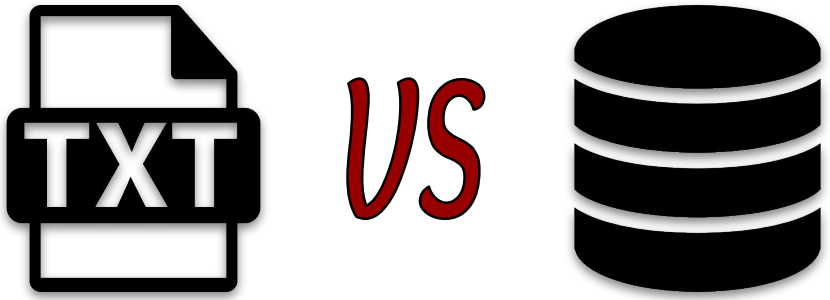
*Dans des **fichiers textes** (CSV, …) ?*

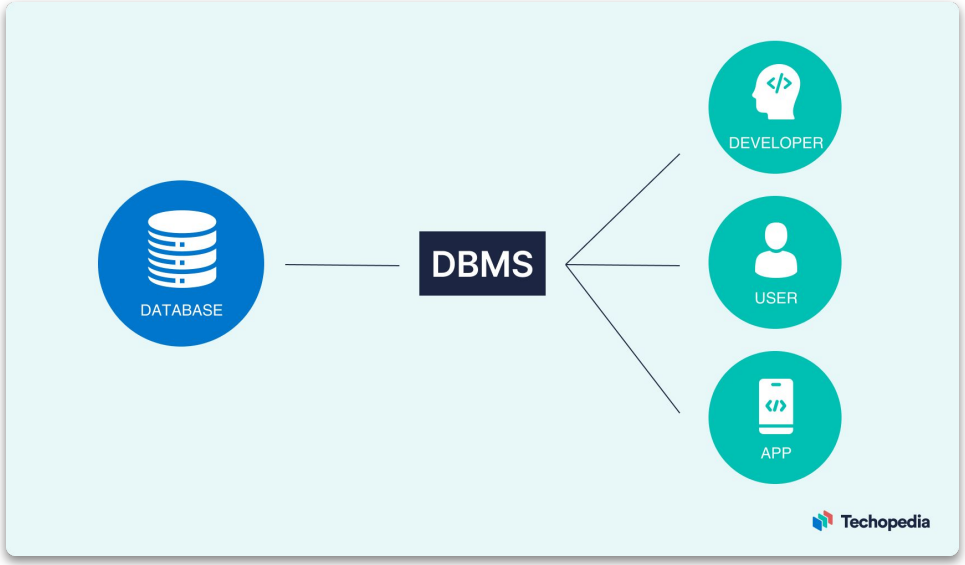
*Dans des **tableurs** (MS Excel, …) ?*

*Dans des **bases de données** (MySQL, …) ?*

Base de données

- Les données peuvent être stockées dans des fichiers textes simples, dans des tableurs, ou dans des bases de données structurées.
- Une **base de données** (***database***) est une collection organisée de données, stockée sur un support informatique et gérée par un système de gestion de base de données (**SGBD**).
- Le SGBD assure la sécurité, la cohérence, l'intégrité, la disponibilité et l'accès concurrent aux données.
- Plusieurs SGBD peuvent coexister et interagir au sein d'un même système d'information (**SI**).





Bases de données relationnelles

- Une **BD relationnelle** (*relational DB*) est un système de stockage de données organisé sous forme de tables (relations) structurées en lignes et en colonnes :
  - Lignes : enregistrements, individus, ou observations.
  - Colonnes : attributs, caractéristiques, ou variables.
- Pour interroger la base de données, on utilise le langage **SQL** (*Structured Query Language*).
- Le **lien** entre deux tables est assuré par un couple de clés :
  - Clé primaire (**PK**) : identifiant unique des enregistrements d'une table.
  - Clé étrangère (**FK**) : clé primaire d'une autre table, utilisée pour établir le lien entre les deux.
- La question qui se pose :

  Comment parvenir à cette structure relationnelle ?

Id	firstName	lastName	birthDate	deptNumber
1009	Montree	Maneevong	04-08-1975	257
1108	Srisuda	Warongkorn	25-04-1985	632
1112	Mana	Tangsrisuk	18-11-1987	598

FK

PK

deptNumber	deptName
257	Accounding
258	Finance
598	IT
632	Marketing

Conception d'une BD relationnelle

- **Identifier** les **éléments** à stocker, les **liens** entre eux, et les **contraintes** à appliquer.
  - Eléments : étudiants, professeurs, cours, formations, …
  - Liens : un étudiant suit des cours, un professeur donne des cours, …
  - Contraintes : un cours est dispensé par **un seul** professeur, …

- **Concevoir** le **Modèle conceptuel de données** (**MCD**) sous forme d'entités et d'associations.
  - Etudiant (identifiant, nom, prénom)
  - Cours (Numéro, intitulé, durée)
  - Professeur (Identifiant, nom, prénom)
  - Associations :
    - Un professeur donne un ou plusieurs cours, et un cours est dispensé par un seul professeur.
    - Un étudiant suit un ou plusieurs cours, et un cours est suivi par plusieurs étudiants.

Formalisme MCD

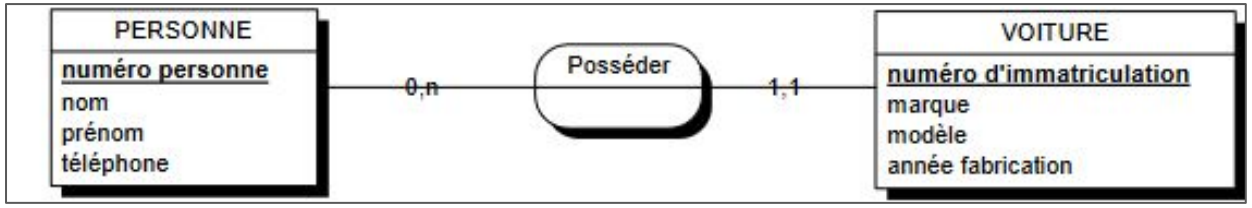
- La **cardinalité** indique le nombre de fois, au minimum et au maximum, qu'une entité participe à une association.
- Les cardinalités traduisent des règles de gestion (contraintes) et peuvent prendre les valeurs suivantes :

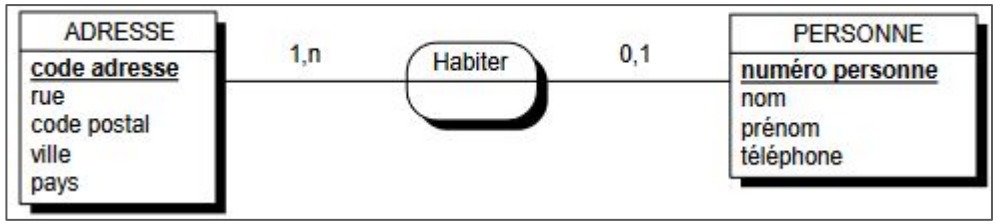
0, 1, n ----------> (0,1) | (1,1) | (0,n) | (1,n)

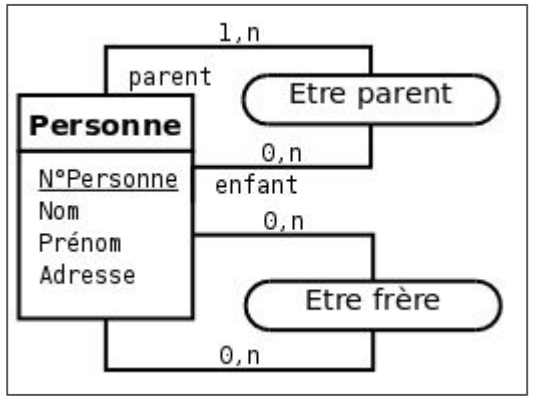
- L'**identifiant** (clé) est un attribut (ou un groupe d'attributs) qui permet d'identifier une entité de manière unique.
- Le **rôle** précise la fonction qu'une entité joue dans une association et fournit des informations supplémentaires sur l'interaction des entités au sein de cette association.

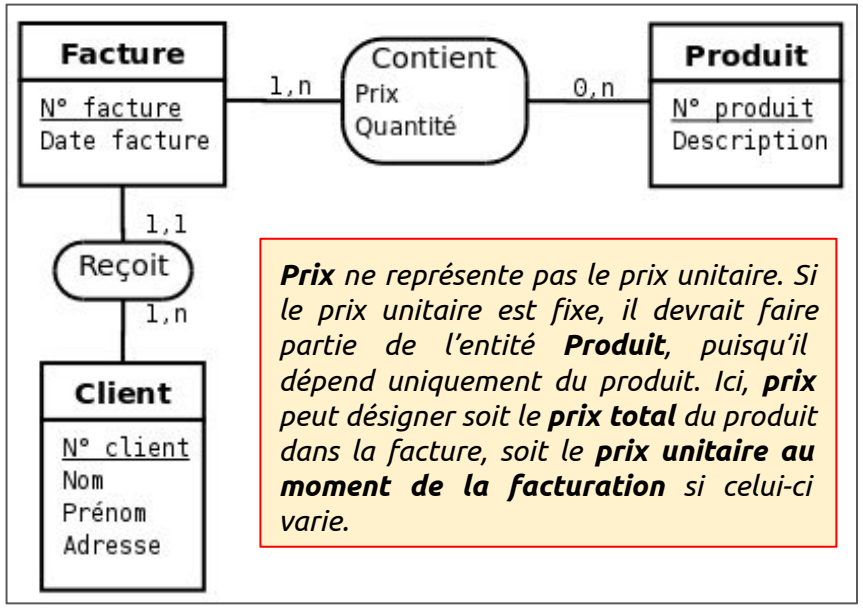
MCD : exemples

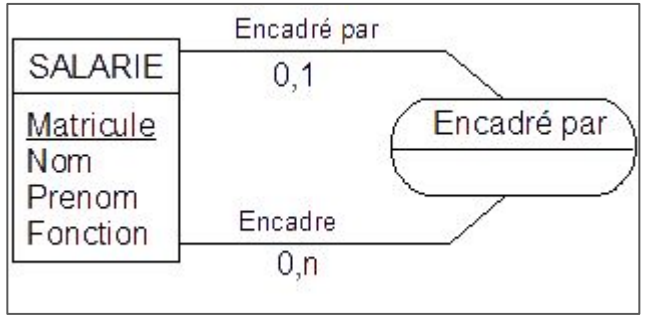
- Si les cardinalités **maximales** des deux côtés sont **1** et **n**, l'association n'a pas besoin de propriétés.

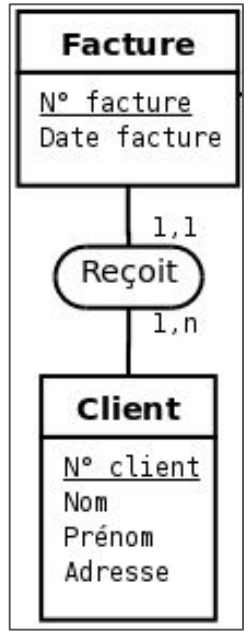






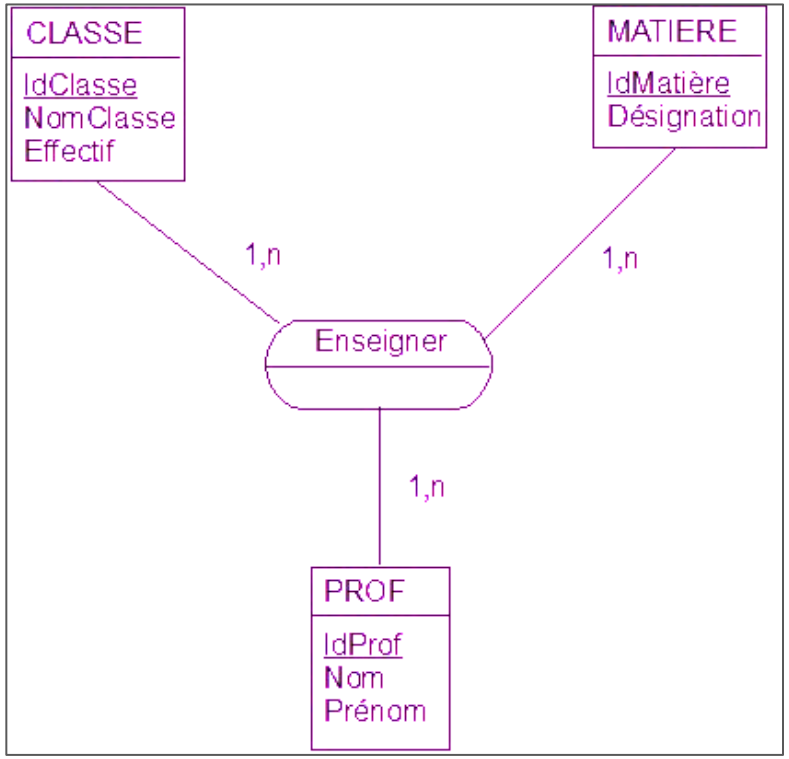


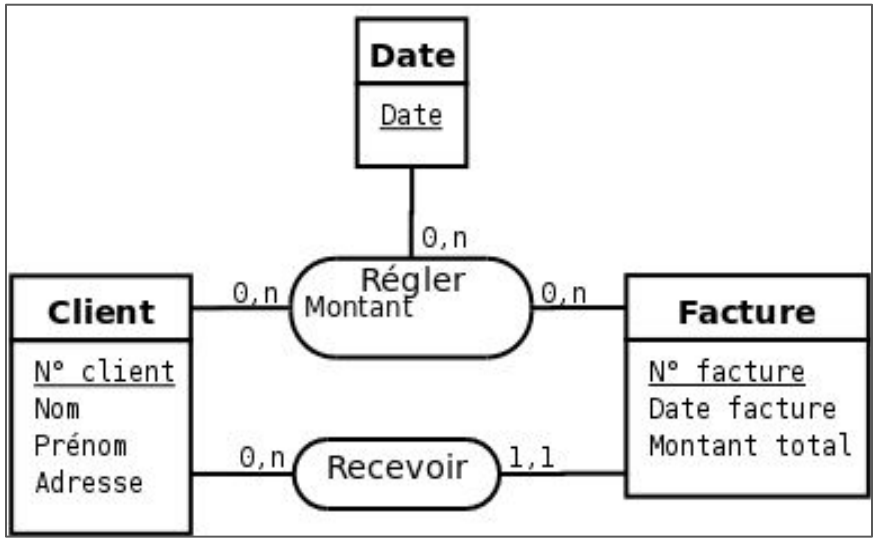


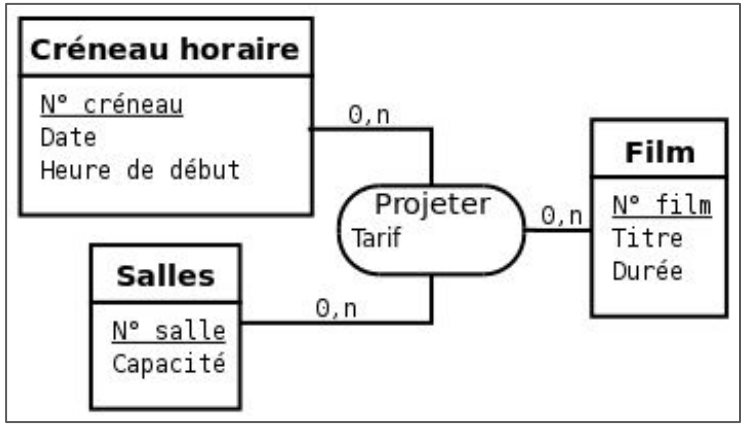


MCD : exemples

- Dans une association **n-aire** :
  - Les cardinalités maximales sont toujours égales à **n**, donc soit (0,n) ou (1,n).
  - Les cardinalités n'ont pas la même signification que dans une association binaire ou réflexive. Elle se lisent comme le nombre minimal et maximal de participations d'une entité à l'association, sans tenir compte des autres entités impliquées.





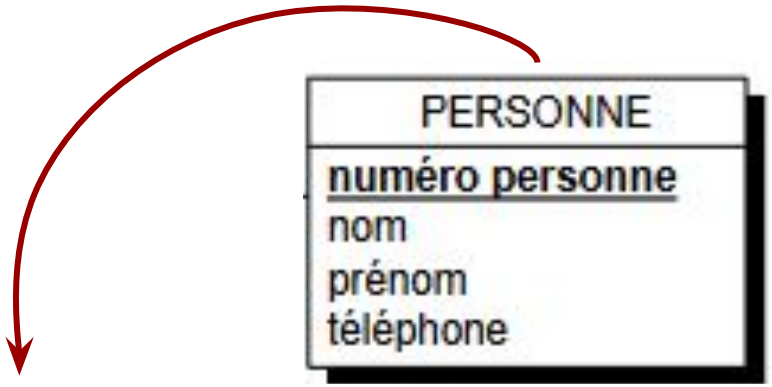


Modèle logique de données

- **Modèle logique de données** (**MLD**), ou modèle relationnel de données, décrit la manière dont les données sont organisées dans une base de données.

- Une table est également appelée **relation**.
- Une **PK** ne peut pas être NULL.

MCD	MLD
Entité	Table (Relation)
Association	Relation
Identifiant	Clé primaire (PK)
-	Clé étrangère (FK)
Propriété	Attribut
Instance	Occurrence



PERSONNE (numero_pers, nom_pers, prenom_pers, tel_pers)

Mauvais MLD

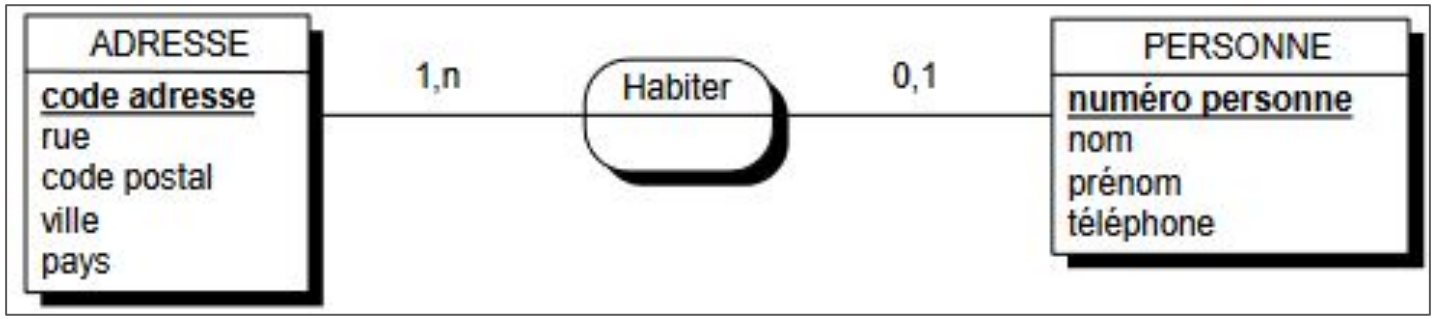
NSS	Nom	Prénom	Immat	Marque	Type	Puiss	Date	Prix
16607...	Dupont	Paul	AJ600AQ	Renault	Clio	5	1/1/96	60000
16607...	Dupont	Paul	AA751KK	Peugeot	504	7	2/7/75	47300
24908...	Martin	Marie	AA751KK	Peugeot	504	7	1/10/89	54900
15405...	Durand	Olivier	AA751KK	Peugeot	504	7	8/8/90	12000
15405...	Durand	Olivier	AJ600AQ	Renault	Clio	5	7/6/98	65000
15405...	Durand	Olivier	XX100XX	BMW	520	10	4/5/01	98000

- **Répétition** de la même donnée : gaspillage de l'espace de stockage.
- Les opérations d'insertion, de mise à jour et de suppression peuvent engendrer des problèmes d'incohérence et de lenteur de traitement.
- Absence de séparation des contextes : par exemple, personne et véhicule sont mélangés.

- Le numéro de sécurité sociale permet d'identifier le nom et le prénom d'une personne.
- Le numéro d'immatriculation permet d'identifier la marque, le type et la puissance d'un véhicule.

MCD → MLD

- Une association (x,1) ←→ (x,n) donne la création d'une #clé étrangère (FK) dans la relation du côté (x,1). La clé étrangère correspond à la clé primaire de l'autre table.
- La clé étrangère DOIT exister au préalable dans l'autre table où elle est clé primaire, ou être NULL si cela est autorisé (cardinalité (**0**,1)).

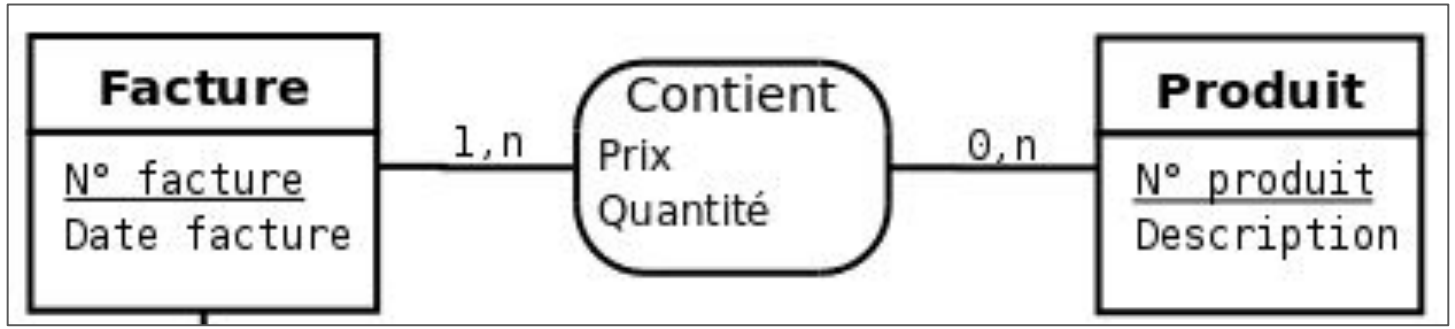


ADRESSE (code_adr, rue, code_postal, ville, pays)

PERSONNE (numero_pers, nom_pers, prenom_pers, tel_pers, #code_adr)

MCD → MLD

- Une association (x,n) ⟷ (x,n) entraîne la création d'une nouvelle relation dont la clé primaire est composée des clés étrangères issues des entités liées par l'association.
- Les propriétés de l'association deviennent alors des attributs de cette nouvelle relation.



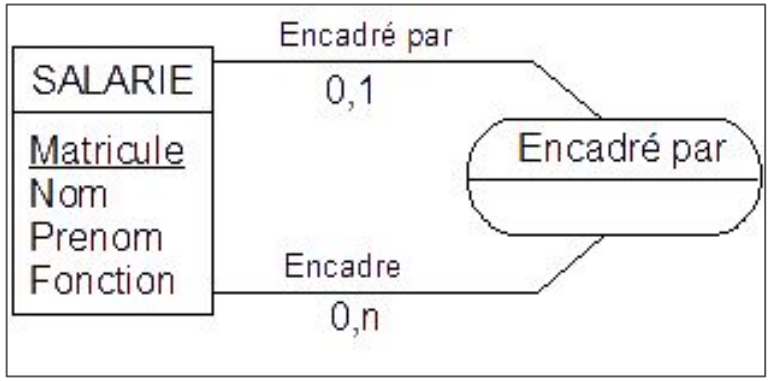
Facture (num_fact, date_fact)

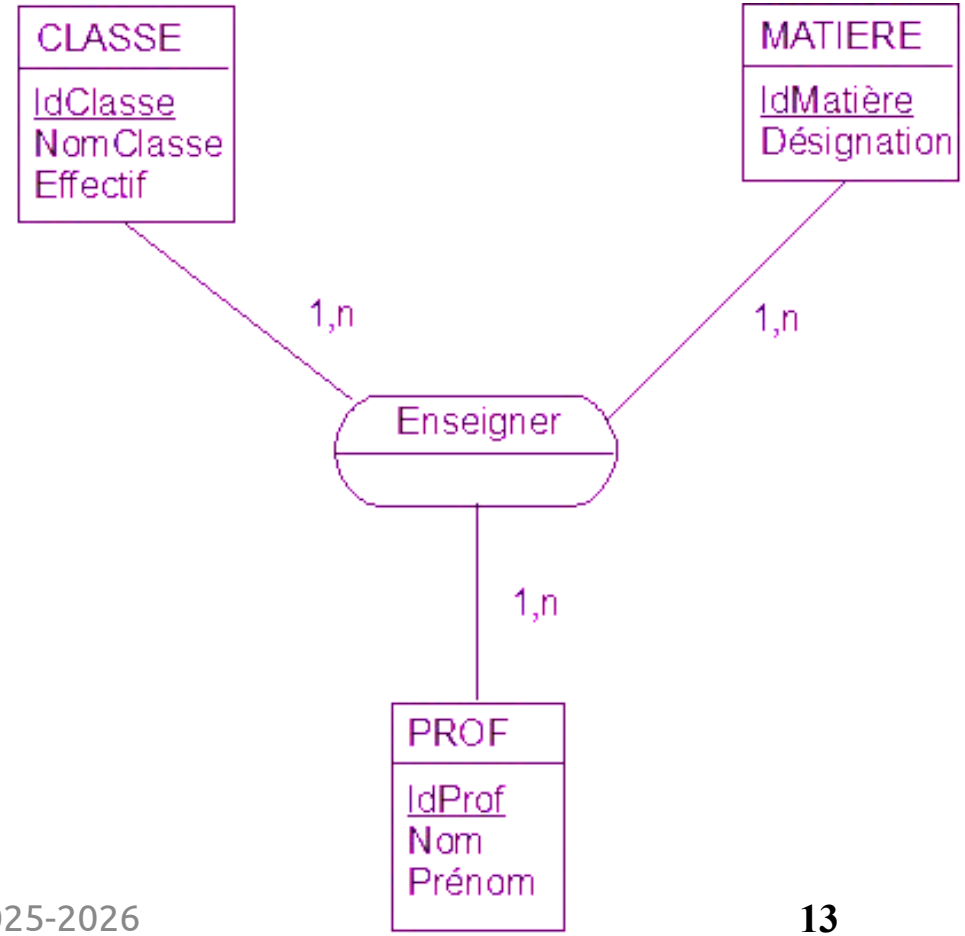
Produit (num_prod, description_prod)

Ligne_facture (#num_fact, #num_prod, prix_prod, quantite_prod)

MCD → MLD

SALARIE (matricule_sal, nom_sal, prenom_sal, fonction_sal,#matricule_encadr)





CLASSE (id_class,nom_class,effectif_class)

MATIERE (id_mat, designation_mat)

PROFESSEUR (id_prof, nom_prof, prenom_prof)

ENSEIGNEMENT (#id_class,#id_mat,#id_prof)

Normalisation

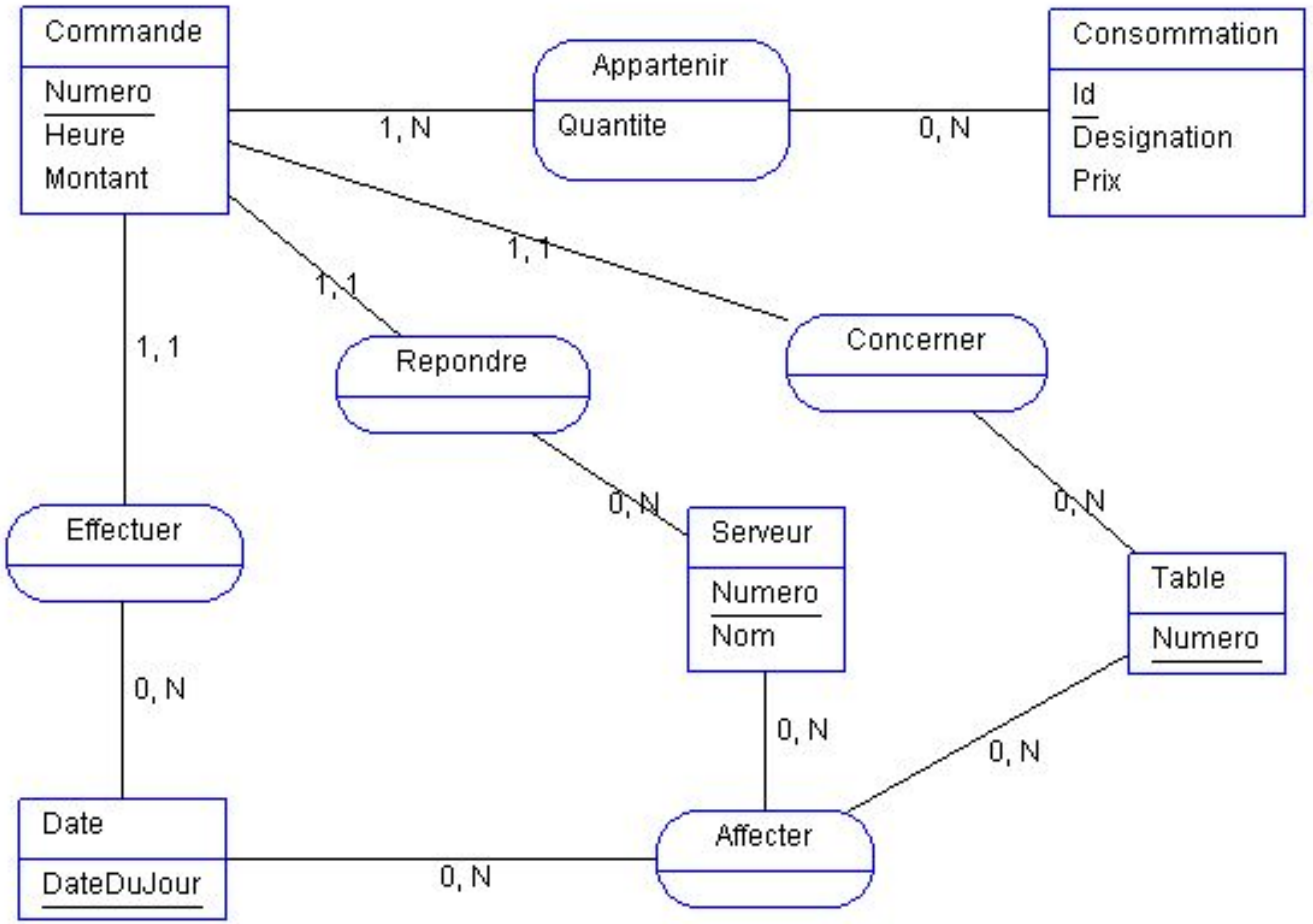
- La normalisation permet d'analyser et de structurer les relations d'une base de données afin de détecter les incohérences et de réduire la redondance des données.
- L'objectif principal est d'optimiser le MLD pour garantir la cohérence, la fiabilité et la facilité de maintenance de la base.

- La normalisation s'effectue à travers une série d'étapes appelées formes normales, où chaque forme normale impose des contraintes supplémentaires par rapport à la précédente :
  - **1FN** : les attributs d'une relation doivent être atomiques, c-à-d ne contenir que des donnée élémentaires.
  - **2FN** : tout attribut non-clé dépend entièrement de la clé primaire, et non d'une partie de celle-ci (pas de dépendance partielle).
  - **3FN** : aucun un attribut non-clé ne doit dépendre d'un autre attribut non-clé.

Exercice 1

Dans une cafétéria, un client s'assoit à une table et peut passer plusieurs commandes de consommations différentes (chacune ayant une désignation et un prix). Pendant une journée, une table est affectée à un unique serveur identifié par son numéro et son nom. On veut connaître, pour le mois précédent, les détails de l'ensemble des commandes effectuées.

Exercice 1 : MCD



Exercice 1 : MLD

Commande (num_com, heure_com, montant_com, #num_serv, #num_tbl, #id_date)

Serveur (num_serv, nom_serv)

Table (num_tbl)

Affectation (#num_serv, #num_tbl, #id_date) // Affecter

Consommation (id_consom, designation_consom, prix_consom)

Comm_Consom (#num_com, #id_consom, quantite) // Appartenir

Date (id_date, date_du_jour)

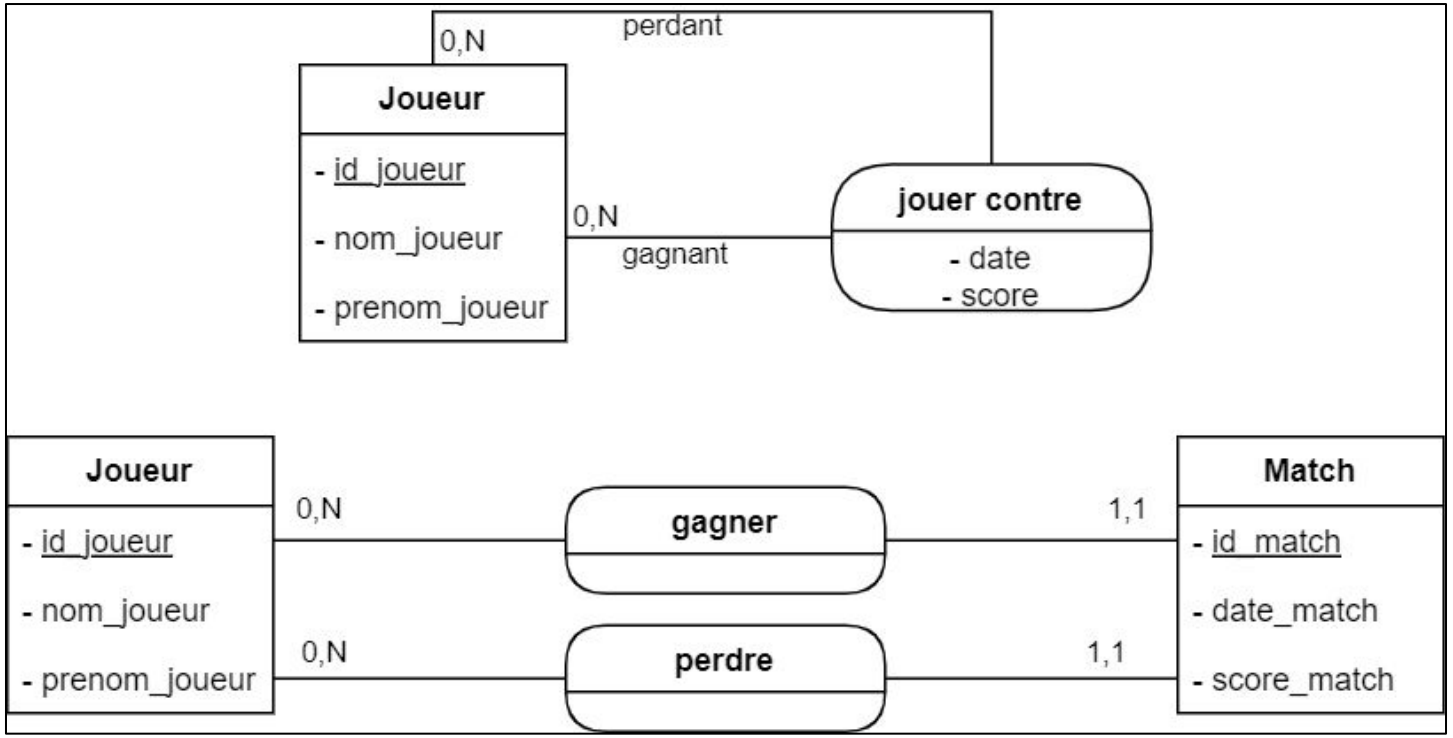
Vérifiez les FN ?

Remarque : il n'est pas nécessaire d'inclure le montant_com, car celui-ci peut être calculé à partir des prix unitaires et des quantités des consommations associées à chaque commande.

Exercice 2

Nous souhaitons concevoir une base de données destinée à enregistrer les noms complets des participants d'un tournoi de tennis ainsi que l'historique complet des matchs disputés. Pour chaque match, nous désirons conserver des informations telles que la date de la rencontre, le nom du joueur vainqueur, ainsi que le score final.

Exercice 2 : MCD



Exercice 2 : MLD

MLD de la première solution avec l'association réflexive :

Joueur (id_joueur, nom_joueur, prenom_joueur)

Rencontre (#id_gagnant, #id_perdant, date, score)

Avec cette solution, deux joueurs ne peuvent pas rejouer un match dans le même tournoi. Si c'est le cas, il faut ajouter le champ "date" à la clé primaire de la table ***Rencontre***.

Ce problème ne se pose pas dans la solution avec deux associations, car un match est identifié par un **ID** unique.

Vérifiez les FN ?

Le MLD de la deuxième solution est :

Joueur (id_joueur, nom_joueur, prenom_joueur)

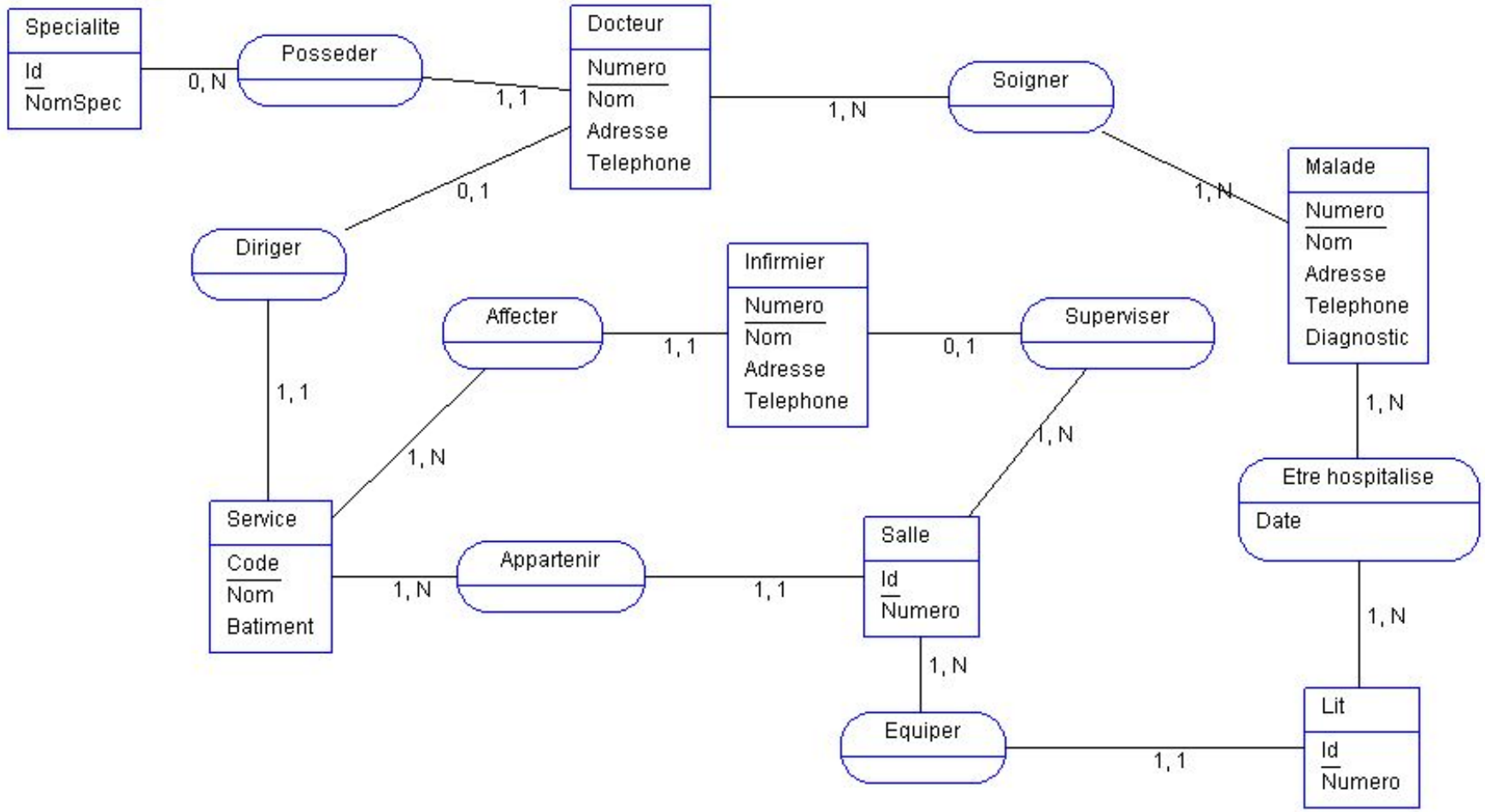
Match (id_match,#id_gagnant, #id_perdant, date_match, score_match)

Exercice 3

Vous êtes chargé de modéliser le système d'information d'un centre de santé. Pour ce faire, vous avez entrepris une analyse approfondie en consultant des documents et en menant des entretiens avec les responsables du centre. Vous avez pu définir les règles suivantes :

- Le centre de santé emploie des médecins et des infirmiers, chacun étant identifié par un numéro d'employé, un nom, une adresse et un numéro de téléphone.
- Le centre est divisé en plusieurs services, chacun ayant un code, un nom, un bâtiment, et est dirigé par un médecin.
- Chaque service comprend plusieurs salles, identifiées par un numéro, supervisées par des infirmiers, et équipées de lits.
- Chaque médecin possède une spécialité, mais n'est pas rattaché à un service spécifique. Par contre, un infirmier est affecté à un seul service.
- Les patients sont hospitalisés dans une salle, avec un numéro de lit et un diagnostic. Ils sont pris en charge par un médecin. En cas de complications, ils peuvent être transférés vers un autre service, dans une autre salle.
- Pour chaque patient, il est essentiel de conserver les informations suivantes : son numéro, son nom, son adresse, son téléphone et sa date d’hospitalisation.

Exercice 3 : MCD



SQL

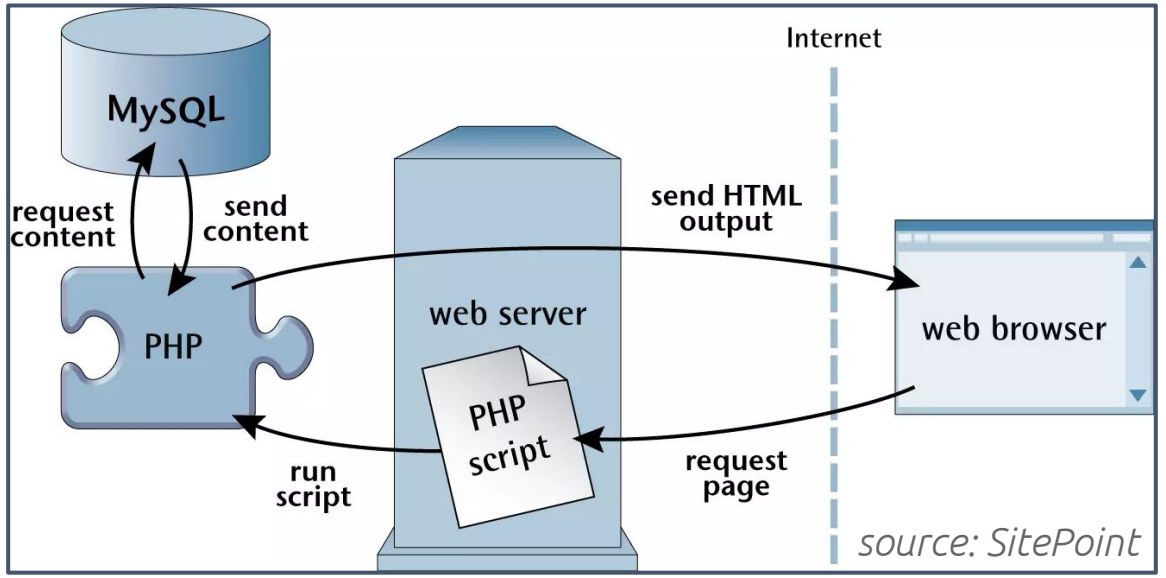
- **SQL** (*Structured Query Language*) est un langage standard pour manipuler et interroger les données dans les bases de données relationnelles à travers des requêtes.

SELECT * FROM Etudiants;

- SQL est un standard mais il peut exister des différences (dialectes, extensions) entre les différents SGBDR.
- Les commandes principales sont classées en 2 ensembles:
  - DML (*Data Manipulation Language*) : SELECT, UPDATE, DELETE, INSERT INTO.
  - DDL (*Data Definition Language*) : CREATE TABLE, ALTER TABLE, DROP TABLE.
- Les commandes SQL sont utilisées pour interagir avec la **structure des tables** et pour manipuler **les données** qu'elles contiennent.
- Les commandes SQL ne sont pas sensible à la casse (select, FROM, update, …), mais les noms d'objets (tables, colonnes, …) peuvent l'être (dans certains SGBDR).
- Il est préférable de terminer chaque commandes SQL par un point-virgule **;**

SQL pour applications

- Pour développer, par exemple, une **application web** qui utilise les données d'une BD, plusieurs composants sont nécessaires :
  - Un **SGBD** (MySQL) → stocker, organiser et gérer les données.
  - Un **langage dynamique côté serveur** (PHP) → exécuter et manipuler les requêtes SQL pour interagir avec la BD.
  - Un **langage d'interface côté client** (HTML/CSS) → structurer les pages web.



MySQL - cmd

```
Microsoft Windows [Version 10.0.19045.6456]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Laatabi>cd "C:\Program Files\MySQL\MySQL Server 8.0\bin"

C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.43 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

SQL

- Créer une base de données : CREATE DATABASE nom_bd;
- Créer une table (avec ou sans contraintes sur les colonnes) :

  CREATE TABLE nom_table (colonne1 type [contrainte], colonne2 type [contrainte], …)

```
mysql> create database jobintech;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| jobintech          |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.01 sec)

mysql> create table Etudiants (CIN varchar(10) primary key, nom varchar(30), prenom varchar(30), date_naissance date);
ERROR 1046 (3D000): No database selected
mysql> use jobintech;
Database changed
mysql> create table Etudiants (CIN varchar(10) primary key, nom varchar(30), prenom varchar(30), date_naissance date);
Query OK, 0 rows affected (0.04 sec)
```

SQL - select

- show databases; *-- afficher toutes les bases de données*
- use nom_db; *-- utiliser la bd "nom_bd"*
- show tables; *-- afficher toutes les tables de la base "nom_db"*
- select * from etudiants; | select nom, prenom from etudiants;
- select * from etudiants order by date_naissance asc, nom desc;
- select distinct prenom from etudiants; | select distinct nom, prenom from etudiants;
- select * from etudiants where prenom = 'Franz';
- select * from etudiants where prenom like 'Fra%' or date_naissance > '2000-01-01';
- select * from etudiants where not (prenom like 'Fra%' and date_naissance > '2000-01-01');
- select * from etudiants where prenom in ('Apache','Franz');
- select * from etudiants where date_naissance between '1900-01-01' and '2000-01-01';

SQL - insert

- insert into etudiants (CIN, nom, prenom, date_naissance)
  values ('HH56', 'Karl', 'Marx', '1880-01-18');
  ⇔ insert into etudiants values ('HH56','Karl','Marx','1880-01-18');
- insert into etudiants (CIN, nom, prenom) values ('AE44','Marie','DesChamps');
- insert into etudiants (date_naissance,CIN,nom) values ('2000-06-08','RR567','Technicien');
- insert into etudiants values ('H61','Kar','Max','1880-01-18'), ('A24','Ma','Des','2000-01-18');

Les colonnes non mentionnées dans la requête INSERT INTO prennent la valeur **NULL** (différente de 0 ou ''), sauf s'il ya une contrainte NOT NULL ou une valeur par DEFAULT.

- select * from etudiants where date_naissance is null;
- select * from etudiants where date_naissance is not null;

→ éviter la comparaison avec **null** : select * from etudiants where date_naissance **=** null;
→ une clé primaire ne peut pas être null.

SQL - update - delete - alter table

- update etudiants set prenom = 'Sup' where CIN='RR567';
- update etudiants set nom = 'Technicien', prenom = 'Sup' where CIN like 'RR%';

- delete from etudiants where CIN='RR567';
- delete from etudiants ; *-- supprimer tous les enregistrements de la table*
- truncate table etudiants; *-- supprimer tous les enregistrements de la table, plus rapide*
- drop table etudiants; *-- supprimer la table elle-même*

- alter table etudiants add taille integer;
- alter table etudiants drop column taille;
- alter table etudiants add taille float default 1.5;
- alter table etudiants rename column taille to taille_m;
- alter table etudiants modify column taille_m double;
- alter table etudiants modify column taille_m double not null;

SQL - contraintes

- Les contraintes sont à spécifier lors de la création (**create**) ou lors de la modification (**alter**) d'une table :

- NOT NULL : la colonne n'accepte pas de valeur NULL.
- UNIQUE : toutes les valeurs de la colonne (ou des colonnes) doivent être uniques.
- PRIMARY KEY : identifie de manière unique une ligne (par nature UNIQUE et NOT NULL).
- FOREIGN KEY : relie une (ou plusieurs) colonnes à la clé primaire d'une autre table.
- DEFAULT : définit une valeur par défaut à une colonne (si non fournie à l'insertion).
- AUTO_INCREMENT : incrémenter automatiquement un entier à chaque insertion (souvent pour les IDs).

```
CREATE TABLE ma_table (
   ID int PRIMARY KEY AUTO_INCREMENT,
   nom varchar(100) NOT NULL,
   age int, taille float, date_naissance date, ID_formation int,
   CONSTRAINT unique_1 UNIQUE (nom, date_naissance),
   CONSTRAINT fk_1 FOREIGN KEY (ID_formation) REFERENCES formation(form_id));
```

SQL - select - jointure

- select count(*) as gen_z from etudiants where date_naissance > '2000-01-01';
- select nom, count(*) as nombre from etudiants group by nom having nombre > 1;
- select nom, avg(taille_m) as moy_taille from etudiants group by nom having count(nom)> 1;
- select nom, avg(taille_m) as moy_taille from etudiants group by nom limit 2;

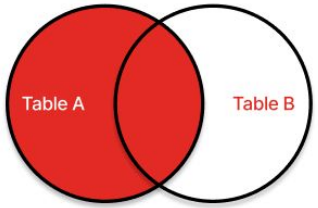
- select etu.nom, prenom, frm.nom from etudiants as etu join formation as frm on etu.id_formation = frm.id;
- *select etu.nom, prenom, frm.nom from etudiants as etu, formation as frm where etu.id_formation = frm.id;*
- *select etu.nom, prenom, frm.nom from etudiants as etu, formation as frm where etu.id_formation = frm.id and frm.heures > 30;*
- *select frm.nom, count(*) as nombre from etudiants as etu, formation as frm where etu.id_formation = frm.id group by frm.nom;*

SQL - jointures

- select etu.nom, prenom, frm.nom
  from etudiants as etu
  join formation as frm
  on etu.id_formation = frm.id
  where frm.heures > 30;
- select frm.nom, count(*) as nombre
  from etudiants as etu
  join formation as frm
  on etu.id_formation = frm.id
  group by frm.nom;
- select sum(frm.heures) as duree from formation as frm;
- select sum(frm.heures) as duree from formation as frm
  join etudiants as etu
  on frm.id=etu.id_formation;
- select * as duree from formation
  join etudiants on
  formation.id=etudiants.id_formation;
- select etu.nom, sum(frm.heures) as duree from formation as frm
  join etudiants as etu
  on frm.id=etu.id_formation
  where date_naissance is not null
  group by etu.nom;

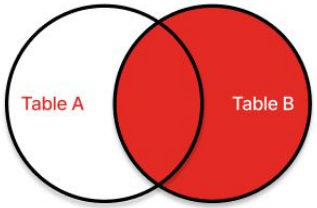
JOINTURES

LEFT JOIN



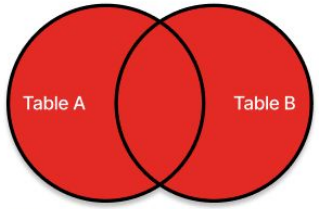

```
SELECT * FROM A LEFT JOIN B ON A.key = B.key;
```

RIGHT JOIN



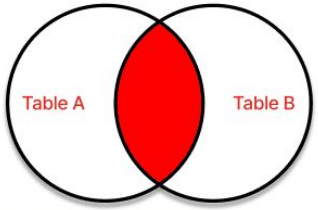

```
SELECT * FROM A RIGHT JOIN B ON A.key = B.key;
```

FULL OUTER JOIN



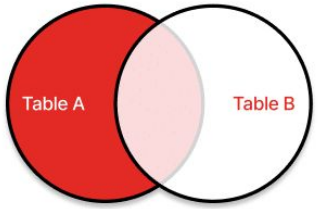

```
SELECT * FROM A FULL OUTER JOIN B ON A.key = B.key;
```

INNER JOIN



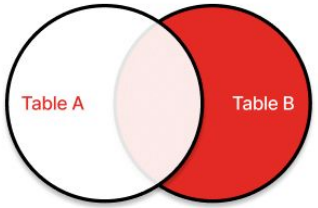

```
SELECT * FROM A INNER JOIN B ON A.key = B.key;
```

LEFT JOIN excluding INNER JOIN



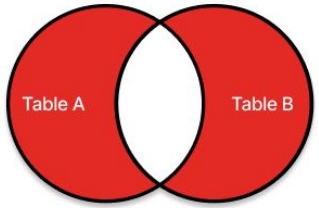

```
SELECT * FROM A LEFT JOIN B ON A.key = B.key
WHERE B.key IS NULL;
```

RIGHT JOIN excluding INNER JOIN



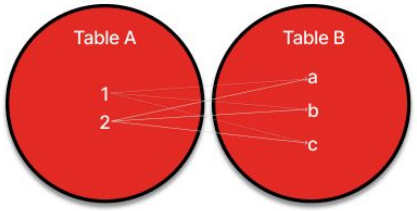

```
SELECT * FROM A RIGHT JOIN B ON A.key = B.key
WHERE A.key IS NULL;
```

FULL OUTER JOIN excluding INNER JOIN




```
SELECT * FROM A FULL OUTER JOIN B ON A.key = B.key
WHERE A.key IS NULL OR B.key IS NULL;
```

CROSS JOIN




```
SELECT * FROM A CROSS JOIN B;
```

source : red9