



Bases de données Relationnelles

ESGF
Un cours de Yann FORNIER

Présentation

Yann FORNIER (28 ans)

Ingénieur en Aérospatial

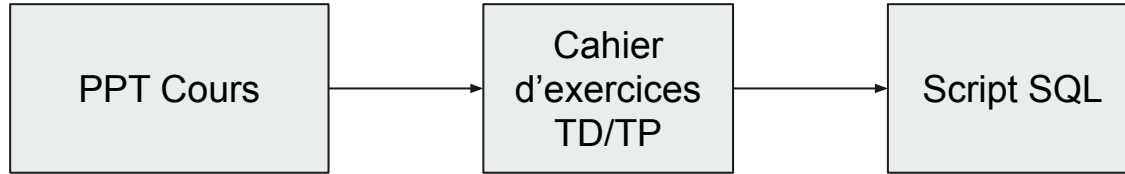
Investisseur dans la finance traditionnelle et dans la DeFi

Enseignant dans l'enseignement supérieur (Informatique)





Architecture du cours et des fichiers





Cours 1

PRÉSENTATION DES BASES DE DONNÉES



Cours 1

Présentation des bases de données, contexte, utilisation.

Concepts d'entité Association, utilisation de clé primaire et clé étrangère

TD : Décomposition d'un énoncé en modèle entité association

Passage d'un modèle Entité Association à un modèle relationnel.



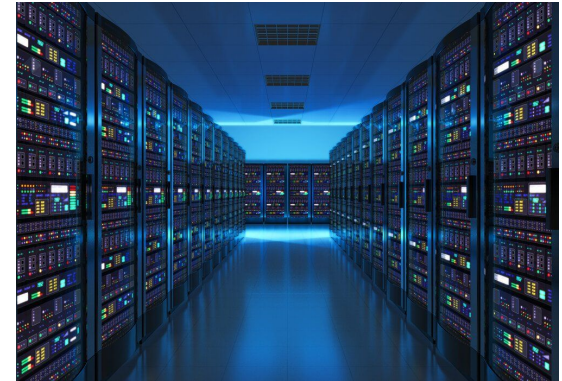
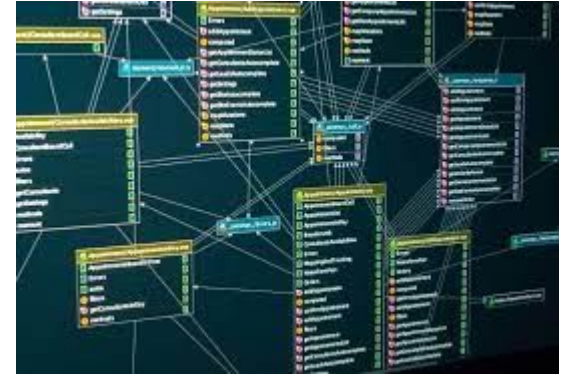
Introduction

Les “BDD” ou “database” dans leur appellation commune, ont pour but de stocker, organiser et analyser les données.

Elles désignent une collection d’informations organisées afin de faciliter la consultation de données, leur gestion et leur mise à jour.

Présentation des bases de données

Les bases de données sont aujourd'hui omniprésentes dans le monde des entreprises mais également dans le quotidien de tous.





Historique des bases de données

Le terme de **base de données** est né en 1964 pour désigner une collection d'informations partagées par différents utilisateurs d'un système d'informations militaires.

Années 70 : Création des Bases de Données Réseaux

Ensemble de fichiers reliés par pointeurs

Langage d'interrogation par navigation



Historique des bases de données

Années 80 : Avènement des Bases de données Relationnelles

Relations entre ensemble de données

Langage d'interrogation par assertion logique

Années 90 : Orientation décisionnelle (Data mining, OLAP)

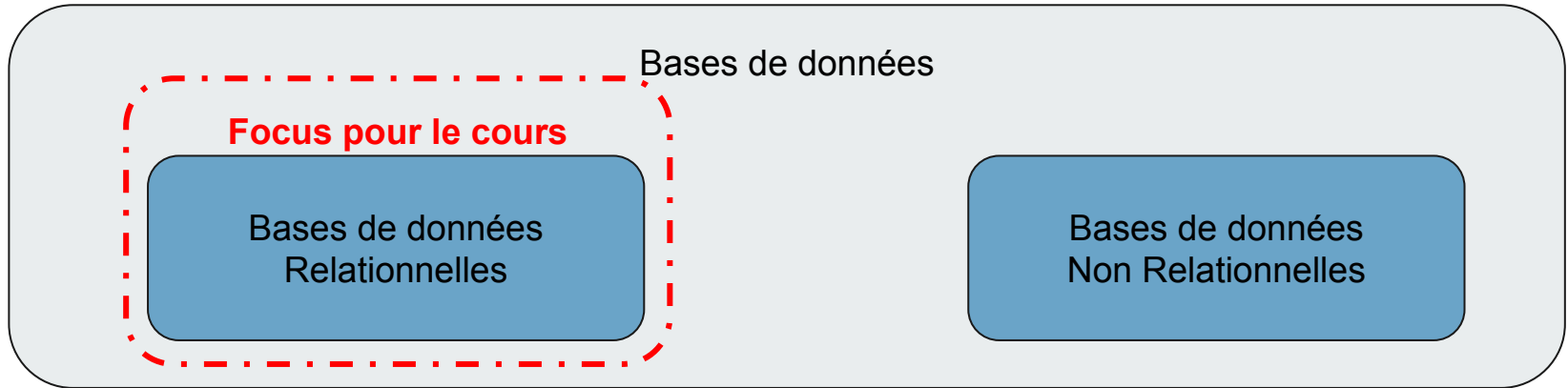
Années 2000 : Avènement du Web

Années 2010 - 2020 : Avènement du Cloud pour les bases de données



Présentation des bases de données

Il existe aujourd'hui 2 grands types de bases de données : **Relationnelles** et **Non Relationnelles**.





Problématique des bases de données

“Pourquoi utiliser plusieurs bases de données reliées entre elles plutôt qu’une grande base de données avec toutes les informations à l’intérieur ?”



Présentation des bases de données relationnelles

Le principal problème des bases de données dites relationnelles provient notamment de la redondance des informations dans une table. Prenons un exemple :

| ID | Nom | Prénom | Délégué | Club1 | Club2 |
|---------|--------|---------|---------|-------|---------|
| 1001245 | Jean | Charles | Dufour | Poker | Finance |
| 1001246 | Dufour | Antoine | Dufour | Photo | Sport |
| 1001247 | Dupont | Marie | Dufour | Poker | Sport |



Présentation des bases de données relationnelles

A l'ajout et à la suppression d'informations d'une base de données, il peut exister des **anomalies**.

Anomalie d'insertion

Anomalie de suppression

Anomalie de modification



Présentation des bases de données relationnelles

Insertion : Si on ajoute un étudiant, ce dernier devra forcément rejoindre 2 clubs.

| ID | Nom | Prénom | Délégué | Club1 | Club2 |
|----------------|----------------|---------------|---------------|----------|----------|
| 1001245 | Jean | Charles | Dufour | Poker | Finance |
| 1001246 | Dufour | Antoine | Dufour | Photo | Sport |
| 1001247 | Dupont | Marie | Dufour | Poker | Sport |
| 1001248 | Azevedo | Pierre | Dufour | ? | ? |

Présentation des bases de données relationnelles

Suppression : Si le dernier étudiant d'un club est supprimé, ce dernier est automatiquement supprimé

| ID | Nom | Prénom | Délégué | Club1 | Club2 |
|---------|--------|---------|---------|-------|---------|
| 1001245 | Jean | Charles | Dufour | Poker | Finance |
| 1001246 | Dufour | Antoine | Dufour | Photo | Sport |
| 1001247 | Dupont | Marie | Dufour | Poker | Sport |

**Le club Photo
disparaît**

Présentation des bases de données relationnelles

Modification : Si une propriété associée à une classe est modifiée, par exemple le délégué

| ID | Nom | Prénom | Délégué | Club1 | Club2 |
|---------|--------|---------|---------|-------|---------|
| 1001245 | Jean | Charles | Dufour | Poker | Finance |
| 1001246 | Dufour | Antoine | Dufour | Photo | Sport |
| 1001247 | Dupont | Marie | Dufour | Poker | Sport |

Modifier le nom
du délégué



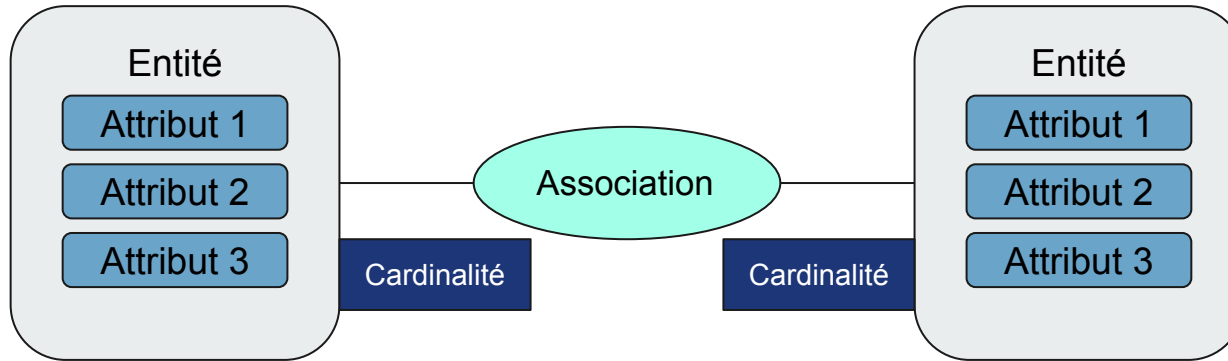
Solution

Pour pallier aux problématiques d'anomalies, on a décomposé les bases de données en plus petits éléments : En Entité et en Association d'entités. Le modèle qui en résulte se nomme :

Le Modèle Entité Association

(ou Modèle Conceptuel de Données (MCD))

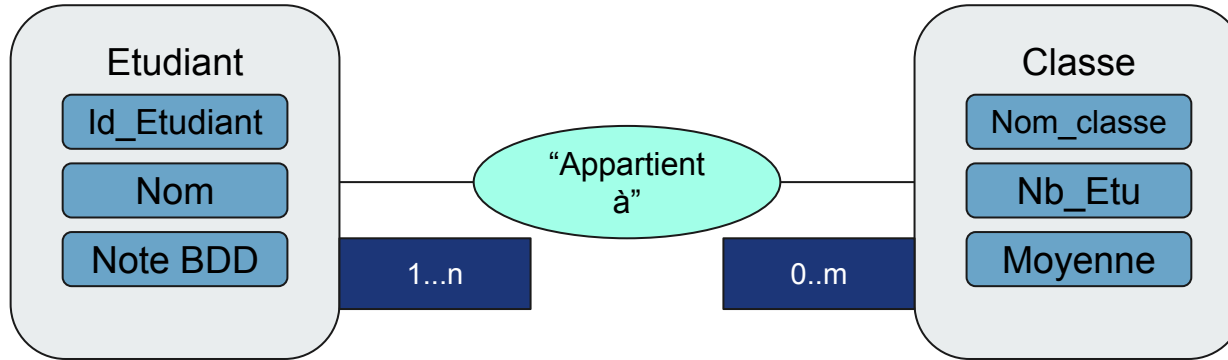
Concept d'Entité Association



Un exemple de modèle Entité Association (E/A)

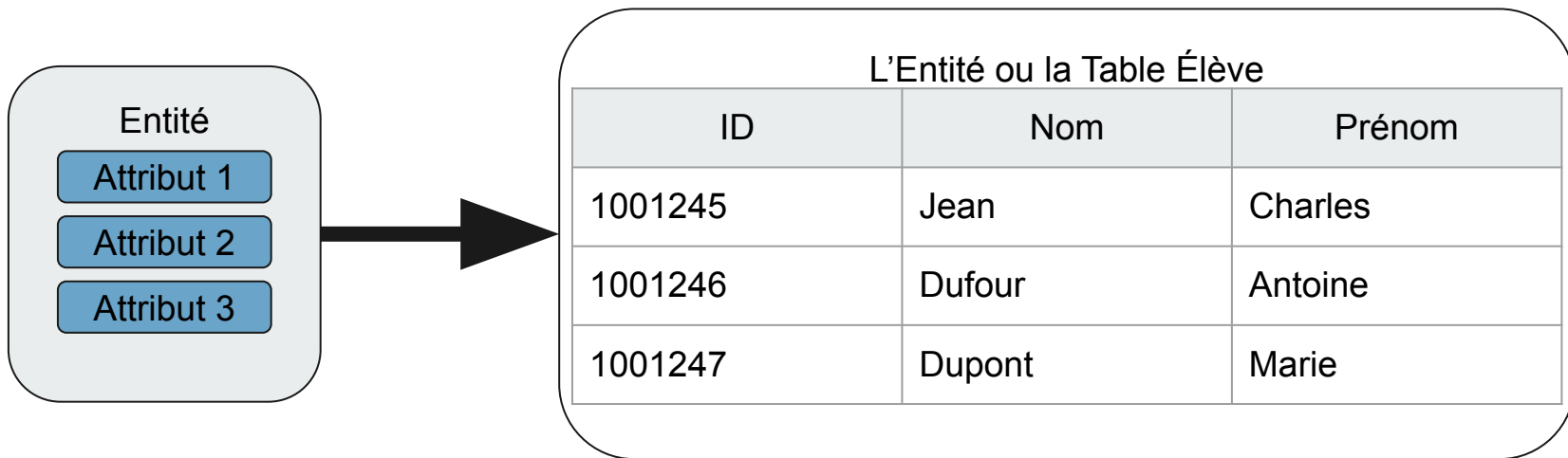
Concept d'Entité Association

Essayons de concrétiser cet exemple...



Concept d'Entité Association

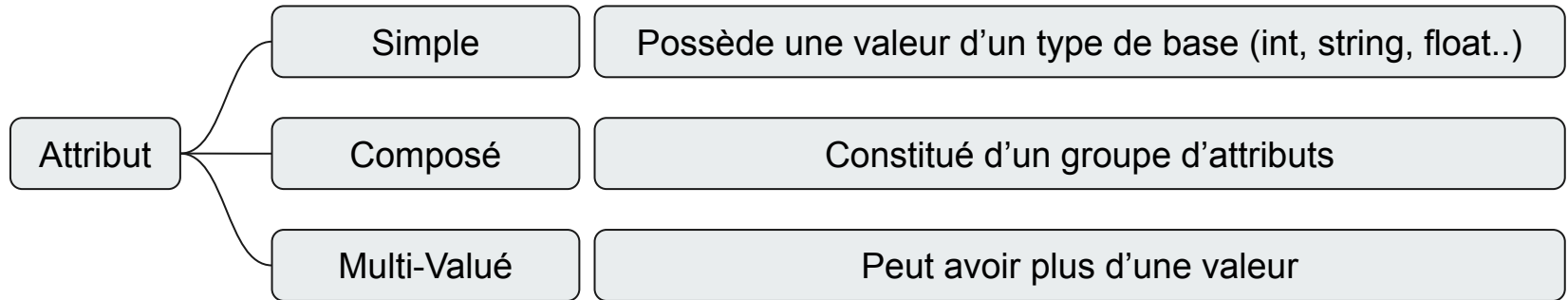
L'entité représente un objet abstrait qui peut contenir un ensemble d'attributs qui lui sont dédiés.





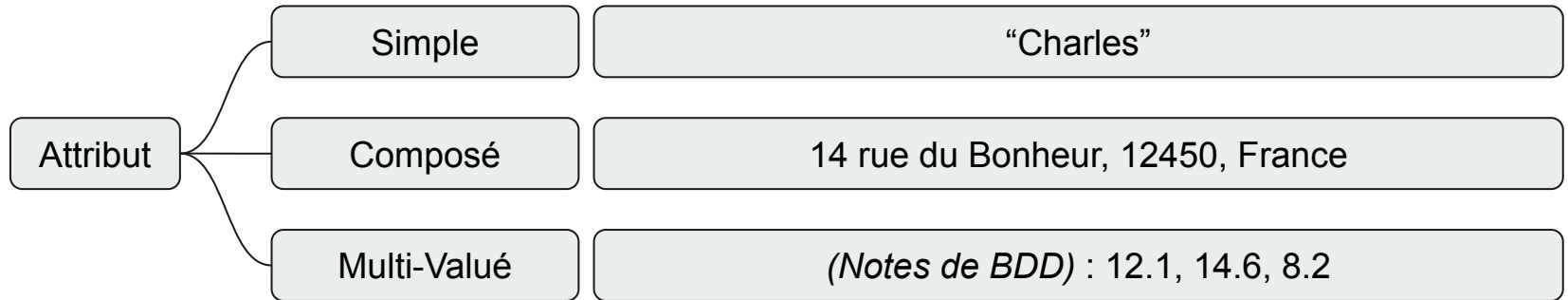
Les attributs dans le modèle Entité Association

Les attributs peuvent être de différentes formes :



Les attributs dans le modèle Entité Association

Les attributs peuvent être de différentes formes :





Concept d'Entité Association

L'association est ce qui va permettre de faire le lien entre 2 bases de données dans le modèle Entité Association. Il prend la forme d'un verbe à l'actif.

“Appartient à”, “possède”, “passe” etc..





Concept d'Entité Association

Lié à l'association, la cardinalité représente un couple de valeurs qui impose une contrainte sur le modèle Entité/Association.

Les cardinalités possibles sont :

1...n

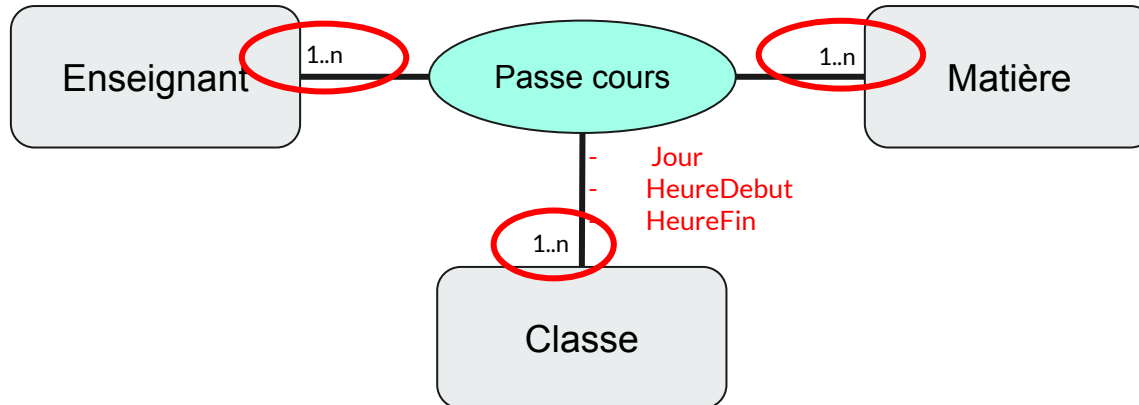
0..1

0...n

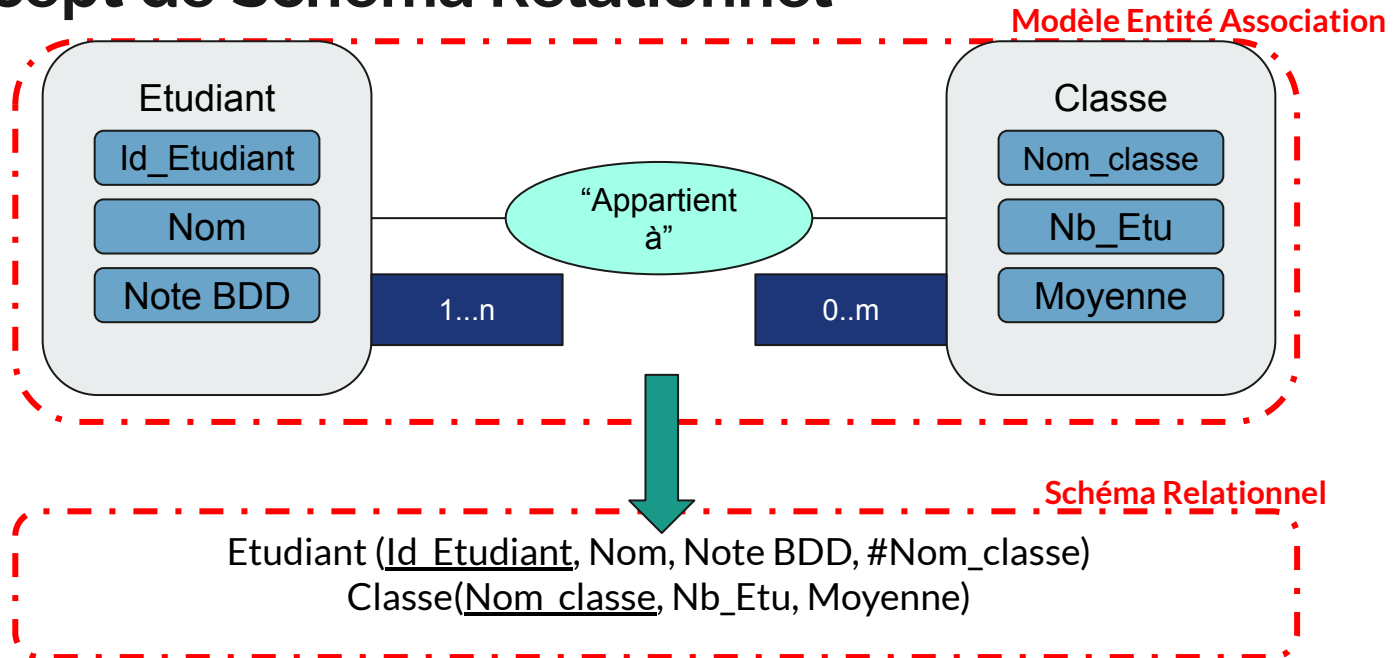
1..1

Association n-aire

Si la cardinalité de chaque entité est identique de chaque côté de l'association, on peut attribuer à l'association une entité.



Concept de Schéma Relationnel



Concept de Schéma Relationnel

Le concept de **Schéma Relationnel** se rapproche davantage de la conception informatique des bases de données

Etudiant (Id_Etudiant, Nom, Note BDD, #Nom_classe)
Classe(Nom_classe, Nb_Etu, Moyenne)



Table Etudiant

| Id_Etudiant | Nom | Note BDD | Nom_Classe |
|-------------|---------|----------|------------|
| 1001245 | Jean | 12.5 | 3A |
| 1002213 | Patrick | 14.1 | 2A |

Table Classe

| Nom_Classe | Nb_Etu | Moyenne |
|------------|--------|---------|
| 3A | 35 | 12.76 |
| 2A | 31 | 13.31 |

Les clés primaires (Identifiants) et clés étrangères

Une clé primaire (appelée aussi “identifiant”) est un attribut qui permet de retrouver une instance unique parmi toutes celles de la table.

Un identifiant peut être constitué de plusieurs attributs.

Etudiant (Id_Etudiant, Nom, Note BDD, #Nom_classe)



Une clé primaire

Maison (N°, Rue, Ville, taille, prix)



Une clé primaire composée

Les clés primaires (Identifiants) et clés étrangères

Les clés étrangères sont des clés primaires qui n'appartiennent pas nativement à l'entité qui la possède.

Elle permet de faire une jointure entre 2 tables.

Etudiant (Id_Etudiant, Nom, Note BDD, #Nom_classe)
Classe(Nom_classe, Nb_Etu, Moyenne)



| Id_Etudiant | Nom | Note BDD | Nom_Classe |
|-------------|---------|----------|------------|
| 1001245 | Jean | 12.5 | 3A |
| 1002213 | Patrick | 14.1 | 2A |

| Nom_Classe | Nb_Etu | Moyenne |
|------------|--------|---------|
| 3A | 35 | 12.76 |
| 2A | 31 | 13.31 |



Les clés primaires (Identifiants) et clés étrangères

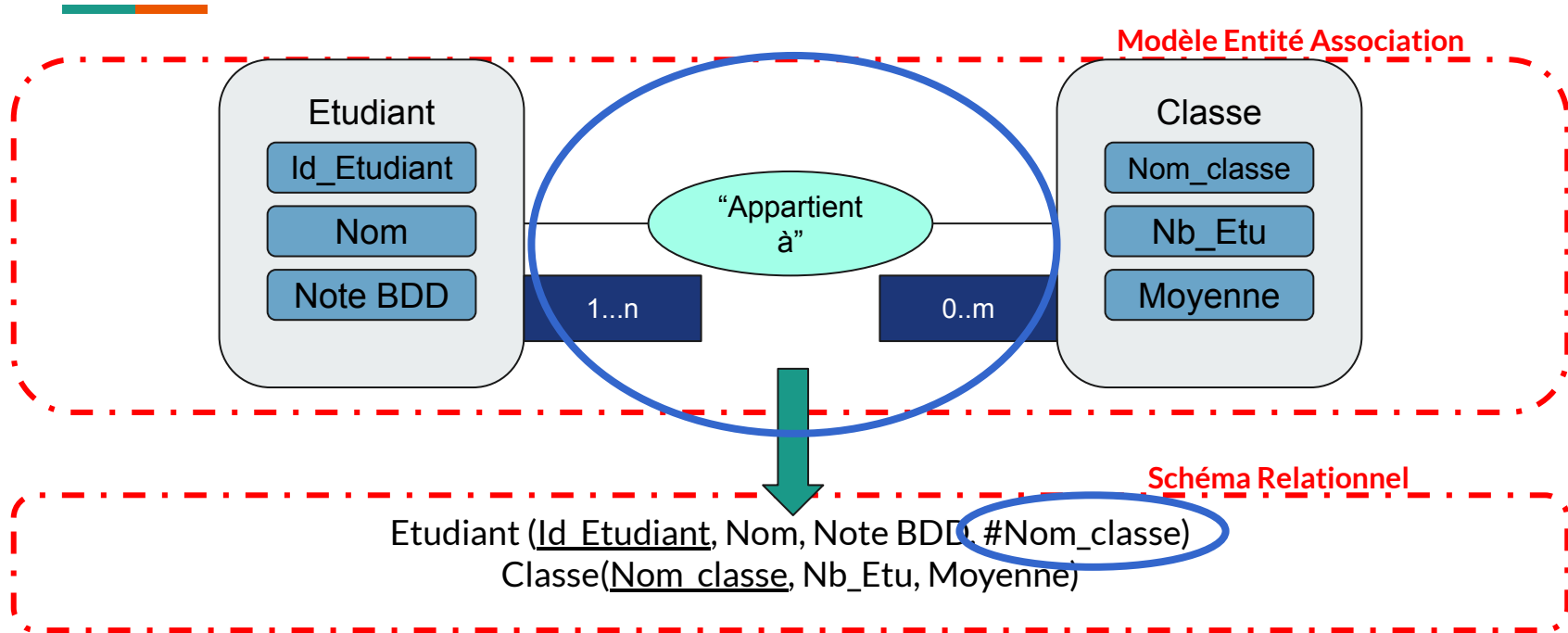
Etudiant (Id_Etudiant, Nom, Note BDD, #Nom_classe)
Classe(Nom_classe, Nb_Etu, Moyenne)

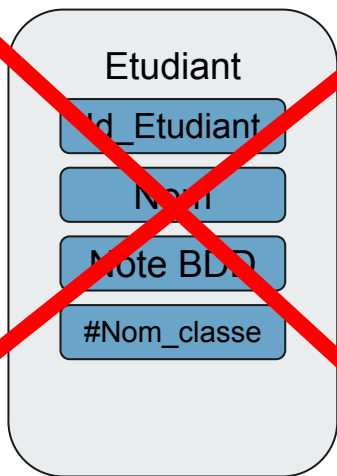
Ici, "Nom_Classe" est la clé primaire de l'entité "Classe" mais est la clé étrangère de l'entité "Etudiant"



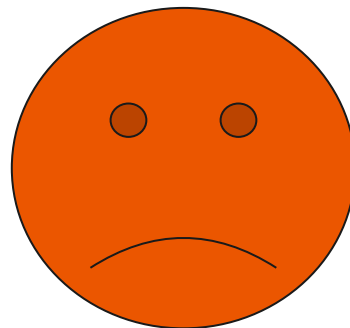
Le modèle Entité Association NE PEUT PAS CONTENIR de clé ÉTRANGÈRES







0/20





Énoncé classique de DST

[Texte descriptif de la situation]

Q1 : “Donner le modèle Entité Association correspondant à la description du texte ci dessus.”

Q2 : “En déduire le schéma relationnel associé”



Exercice d'application

Enoncé : Un client a un numéro de sécurité sociale, un nom, un prénom, un bonus et un malus

Il passe un contrat avec un agent pour chacun de ses véhicules, pour certains risques couverts. Un véhicule est caractérisé par un numéro, une puissance, une marque, un type et une couleur. Les clients ont parfois des sinistres avec des tierces personnes. Les tierces personnes ont un numéro de sécurité sociale et sont assurées auprès d'une compagnie d'assurance. Une compagnie d'assurance a un nom et une adresse. On doit connaître le lieu et la date du sinistre.

Q1 : “Donner le modèle Entité Association correspondant à la description du texte ci dessus.”

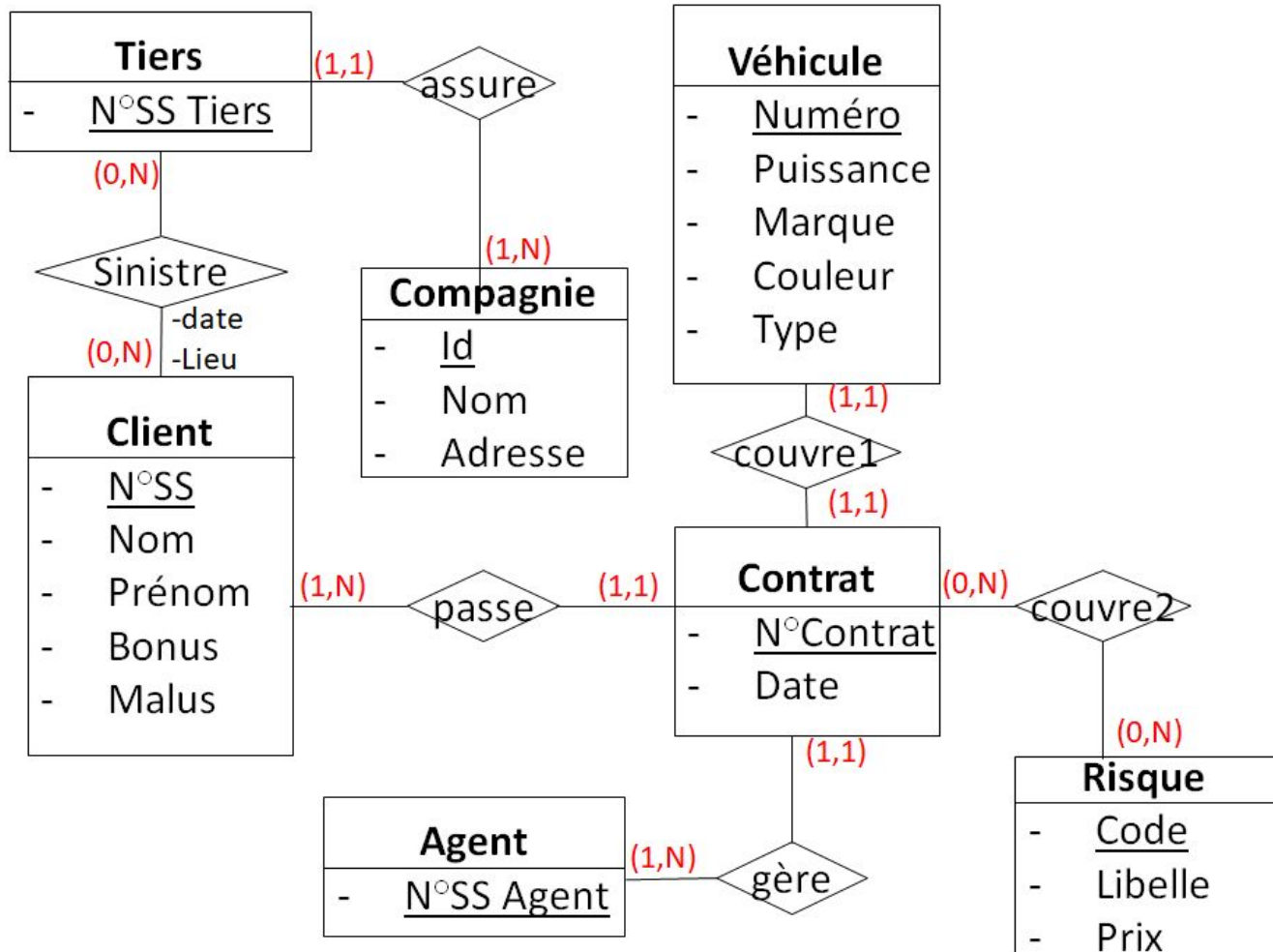
Q2 : “En déduire le schéma relationnel associé”



Exercice d'application

Un **client** a un **numéro de sécurité sociale**, un **nom**, un **prénom**, un **bonus** et un **malus**

Il **pass**e un **contrat** avec un **agent** pour chacun de ses **véhicules**, pour certains **risques couverts**. Un véhicule est caractérisé par un **numéro**, une **puissance**, une **marque**, un **type** et une **couleur**. Les clients ont parfois des **sinistres** avec des **tierces personnes**. Les tierces personnes ont un **numéro de sécurité sociale** et **sont assurées** auprès d'une compagnie d'assurance. Une **compagnie d'assurance** a un **nom** et une **adresse**. On doit connaître le **lieu** et la **date du sinistre**.



Les tables/relations :

- Client (N°SS, Nom, Prénom, Bonus, Malus)
- Contrat (N°Contrat, Date, #N°SS, #Numéro, #N°SS-Agent)
- Agent (N°SS-Agent, Nom, Prénom, Adresse,...)
- Véhicule (Numéro, Puissance, Marque, Couleur, Type, # N°Contrat)
- Risque (Code, libellé, Prix)
- Tiers (N°SS-Tiers,..., #Id)
- Compagnie (Id, Nom, Adresse)
- Sinistre (N°Sinistre, date, lieu, # N°SS, # N°SS-Tiers) *ou* (# N°SS, # N°SS-Tiers, date, lieu)
- Couvre2(N°Couvre, # N°Contrat, # Code) *ou* (# N°Contrat, # Code)



Pour un système d'informations particulier, il n'existe pas de modèle conceptuel unique.

Le bon modèle est celui validé par l'ensemble des membres du projet.



Cours 2

TD1

Exercices modèle Entité Association

Schéma relationnel



TD/TP1

Rendez vous sur le fichier “TD TP BDDR.pdf” et commencez la section 1.



Cours 3

Vérification et Normalisation



Sommaire

Présentation des concepts de Dépendance Fonctionnelle

Normalisation de bases de données - Enjeux et Utilisations

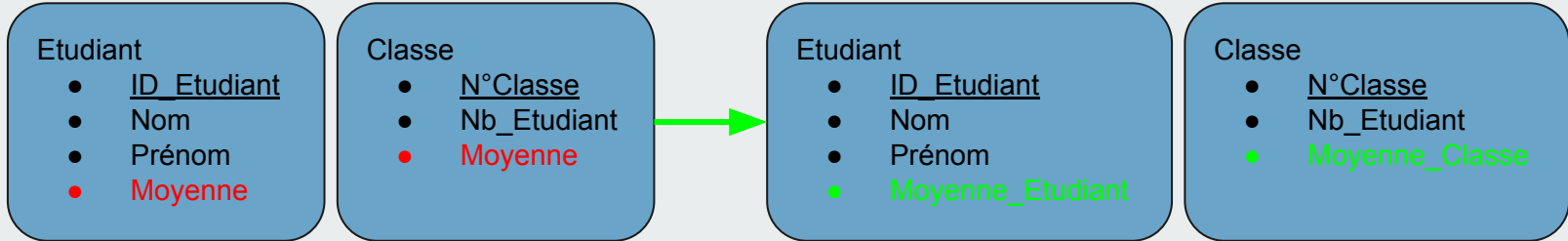
Présentation des Formes Normales 1,2 et 3



Observations

La qualité d'un modèle Entité Association peut être dégradée par plusieurs types de conflits sémantiques liés soit au schéma soit aux données :

Schéma



Données

| Id_Thermistances | Nom | Bus_Data | Port |
|------------------|-----------|----------|-------|
| 1562486 | THERM_001 | CAN_001 | CAN |
| e15 | therm_002 | CAN_001 | 2207 |
| 1378945 | th_003 | can1 | ----- |

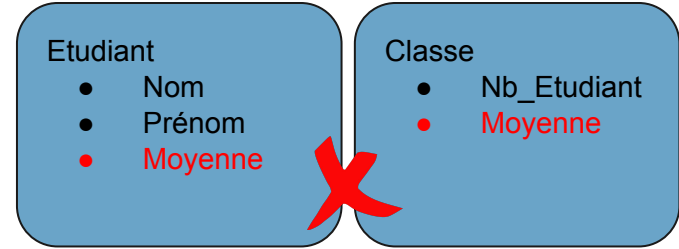


Vérifications et normalisations des modèles

Afin de pouvoir déployer un modèle E/A en base de données relationnelle, on effectue des étapes nécessaires afin de s'assurer de ne pas avoir de mauvaises surprises lors de son déploiement et de son utilisation.

Vérification

- Tout attribut doit apparaître une seule fois dans un modèle.

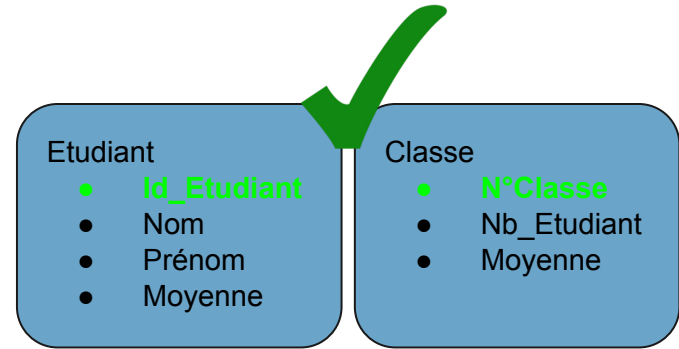


- Toutes les propriétés identifiées doivent apparaître dans le modèle.



Vérification

- Toutes les entités ont un identifiant



- Pas d'héritage dans le modèle E/A de base



Normalisation

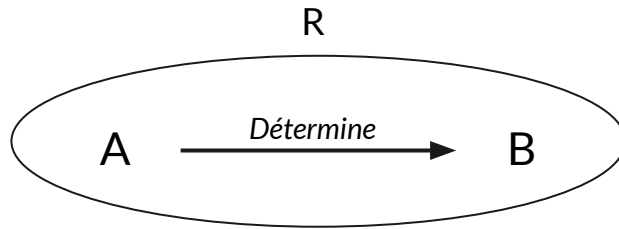
La normalisation a pour but d'éviter la redondance de données dans le modèle.

Pour cela on va définir le concept de Dépendance Fonctionnelle qui va lier les attributs d'une relation.

Qu'est ce qu'une dépendance fonctionnelle ?

Une dépendance fonctionnelle est une contrainte entre 2 ensembles d'attributs dans une table de base de données.

Dans une relation R, on dit qu'un attribut A détermine un attribut B.



On dit également que **B dépend fonctionnellement de A.**





Exemple

| Animal | Type | Poids | Vitesse |
|----------|------------|------------|------------|
| Chien | Vertébré | 8 kg | 30 km/h |
| Chat | Vertébré | 8 kg | 40 km/h |
| Escargot | Invertébré | 0,045 kg | 0,048 km/h |
| Baleine | Vertébré | 130 000 kg | 40 km/h |

Exemple

| | | | |
|----------|------------|----------|------------|
| Animal | Type | Poids | Vitesse |
| Chat | Vertébré | 8 kg | 30 km/h |
| Chat | Vertébré | 9 kg | 40 km/h |
| Escargot | Invertébré | 0,045 kg | 0,048 km/h |
| Gorille | Vertébré | 8 kg | 40 km/h |

Ici {Type, Poids, Vitesse} → Animal



Quelques règles sur les dépendances fonctionnelles

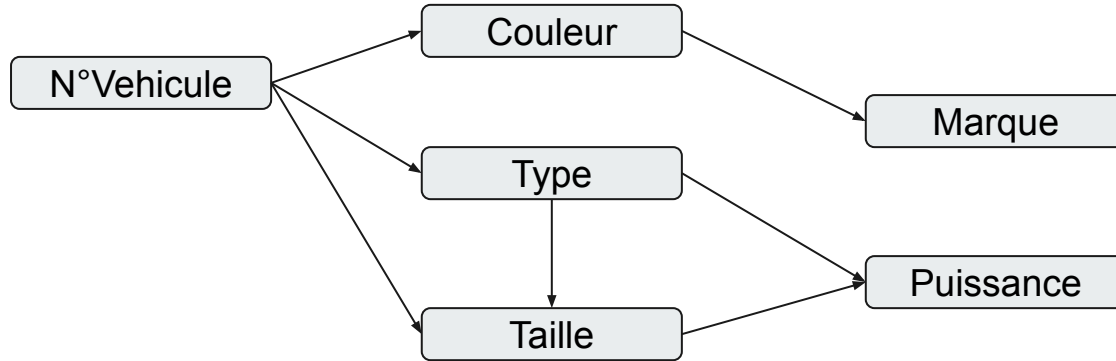
- Tous les attributs d'une entité dépendent fonctionnellement et uniquement de l'identifiant.
- Théorème :
 - Soit $R(A_1, A_2, \dots, A_n)$,
 - X et Y des sous-ensembles d'attributs de R , (ex: $X = A_1, A_2$ et $Y = A_4$)
 - on dit que **X détermine Y** si et seulement si :

$$\exists ! f, f(X) = Y$$

$$\text{ex : } f(A_1, A_2) = A_4$$

Graphe de Dépendance Fonctionnelle

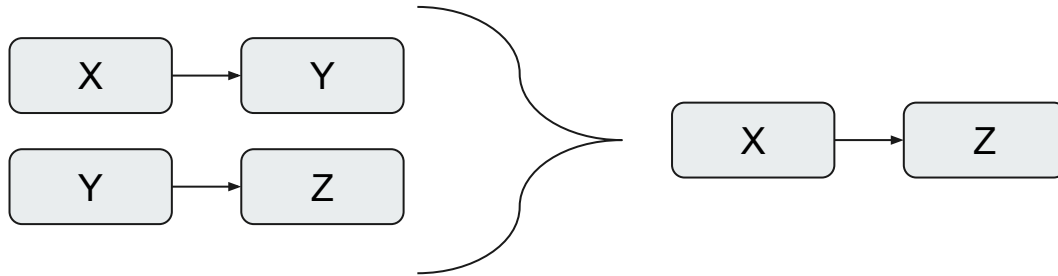
Voiture (N°Vehicule, Type, Couleur, Marque, Puissance, Taille)



Axiomes d'Armstrong

Les Axiomes d'Armstrong permettent de définir des règles entre plusieurs attributs liés par des dépendances fonctionnelles :

- Transitivité





Axiomes d'Armstrong

Les Axiomes d'Armstrong permettent de définir des règles entre plusieurs attributs liés par des dépendances fonctionnelles :

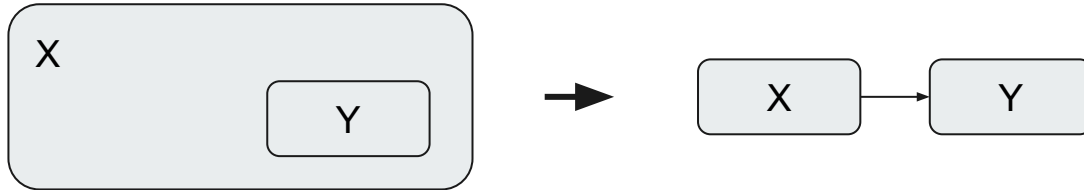
- Augmentation



Axiomes d'Armstrong

Les Axiomes d'Armstrong permettent de définir des règles entre plusieurs attributs liés par des dépendances fonctionnelles :

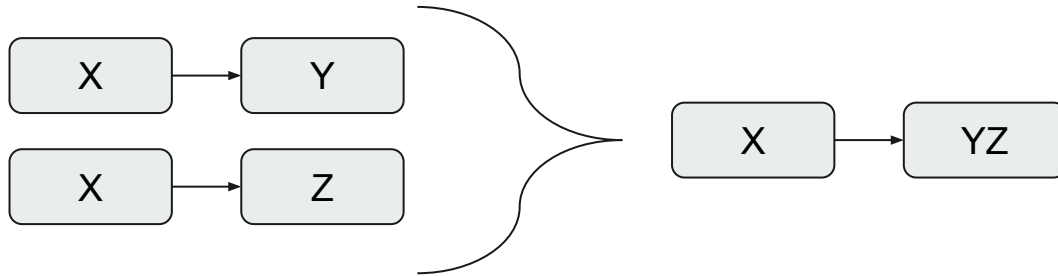
- Réflexivité



Axiomes d'Armstrong

Les Axiomes d'Armstrong permettent de définir des règles entre plusieurs attributs liés par des dépendances fonctionnelles :

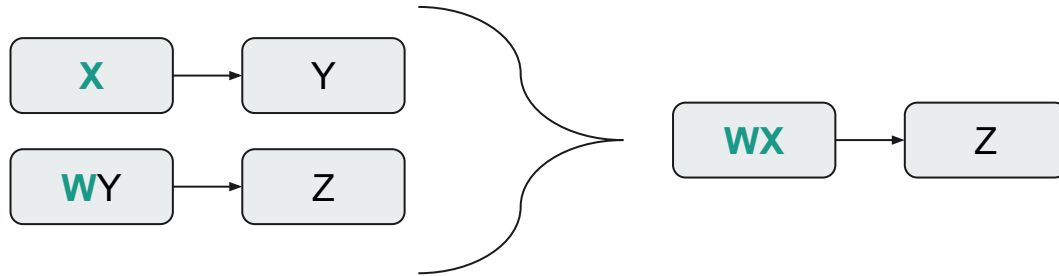
- Union



Axiomes d'Armstrong

Les Axiomes d'Armstrong permettent de définir des règles entre plusieurs attributs liés par des dépendances fonctionnelles :

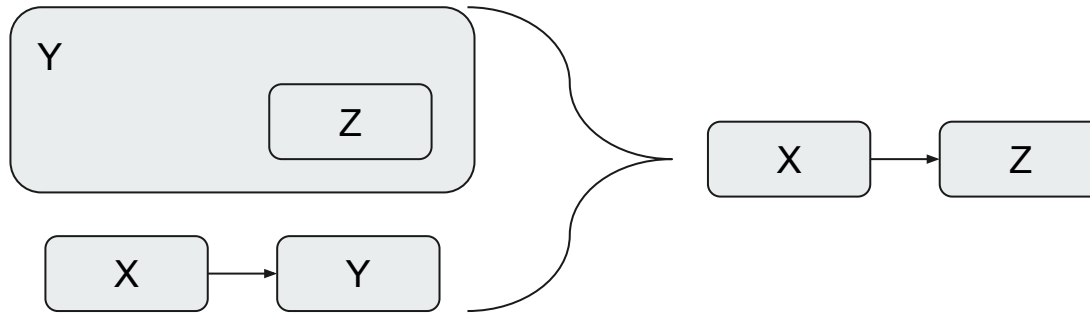
- Pseudo Transitivity



Axiomes d'Armstrong

Les Axiomes d'Armstrong permettent de définir des règles entre plusieurs attributs liés par des dépendances fonctionnelles :

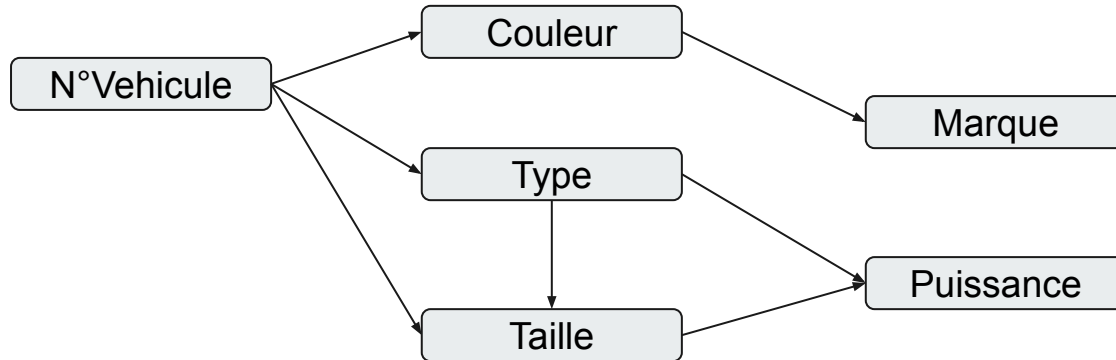
- Décomposition



Détermination de la clé

Voiture (N°Vehicule, Type, Couleur, Marque, Puissance, Taille)

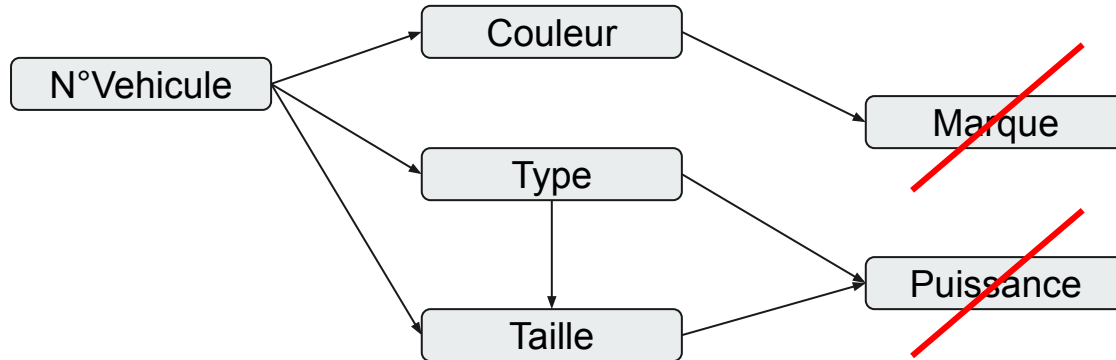
- Tous les attributs d'une entité dépendent fonctionnellement et uniquement de l'identifiant.



Détermination de la clé

Voiture (N°Vehicule, Type, Couleur, Marque, Puissance, Taille)

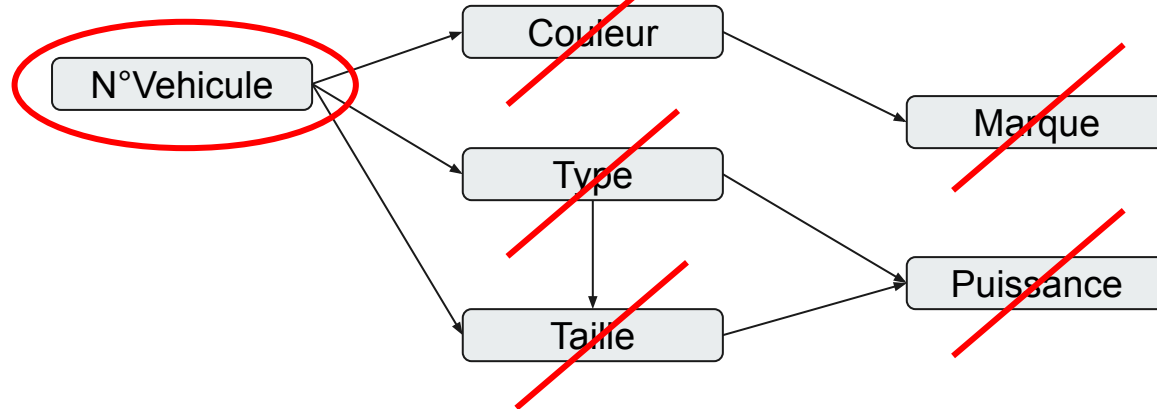
- Tous les attributs d'une entité dépendent fonctionnellement et uniquement de l'identifiant.



Détermination de la clé

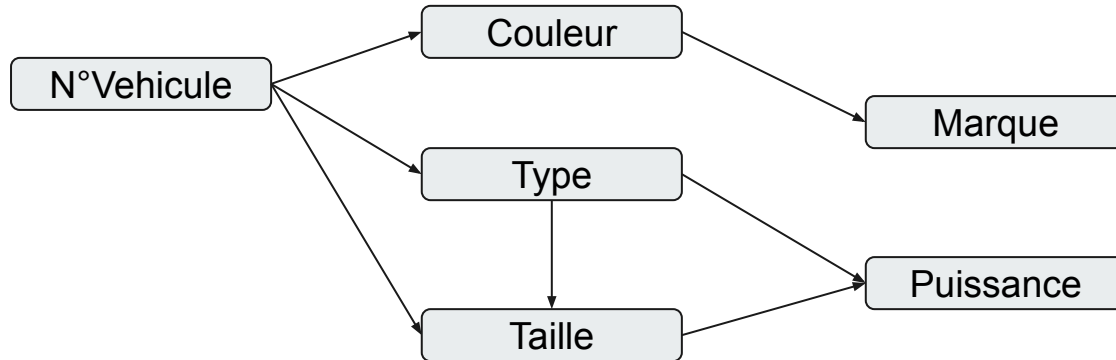
Voiture (N°Vehicule, Type, Couleur, Marque, Puissance, Taille)

- Tous les attributs d'une entité dépendent fonctionnellement et uniquement de l'identifiant.



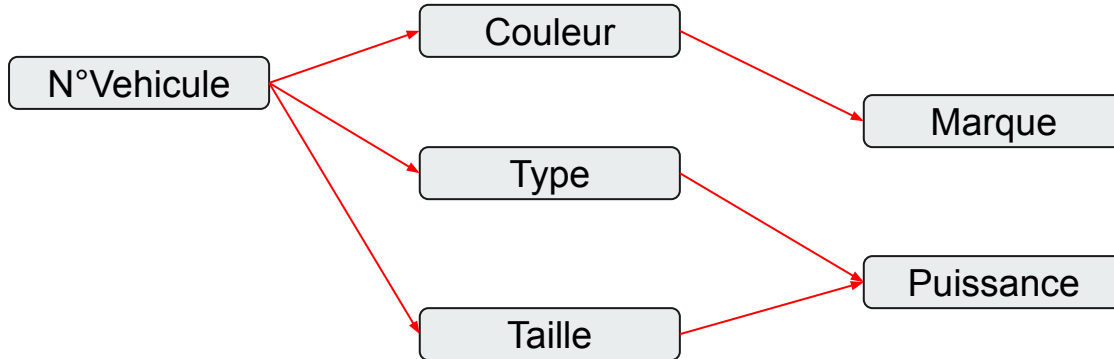
Couverture minimale

La couverture minimale d'un ensemble de DFE (*Dépendance Fonctionnelle Élémentaire*) est un sous ensemble minimum de DFE permettant de générer toutes les autres DFE



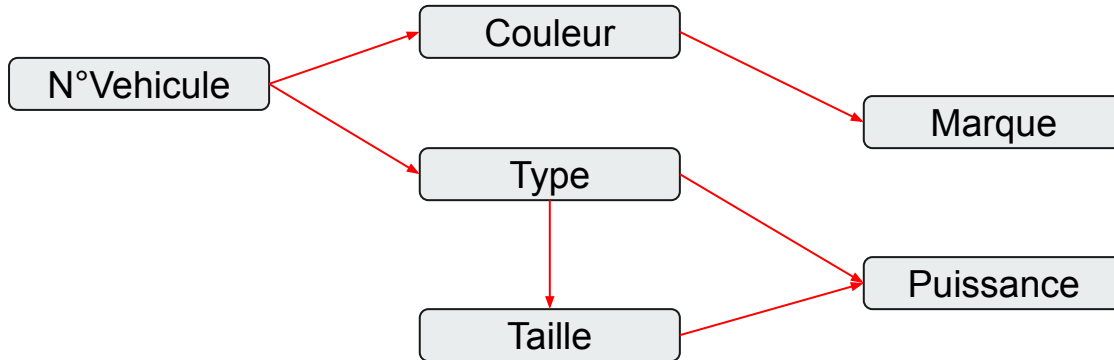
Couverture minimale

La couverture minimale d'un ensemble de DFE (*Dépendance Fonctionnelle Élémentaire*) est un sous ensemble minimum de DFE permettant de générer toutes les autres DFE



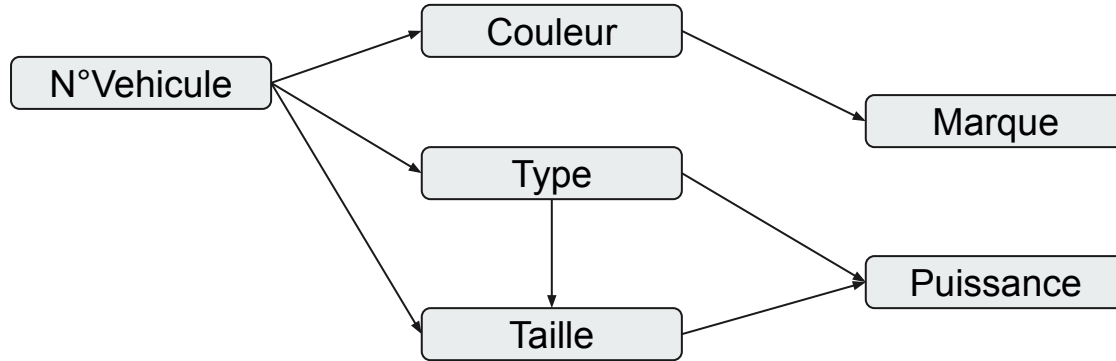
Couverture minimale

La couverture minimale d'un ensemble de DFE (*Dépendance Fonctionnelle Élémentaire*) est un sous ensemble minimum de DFE permettant de générer toutes les autres DFE



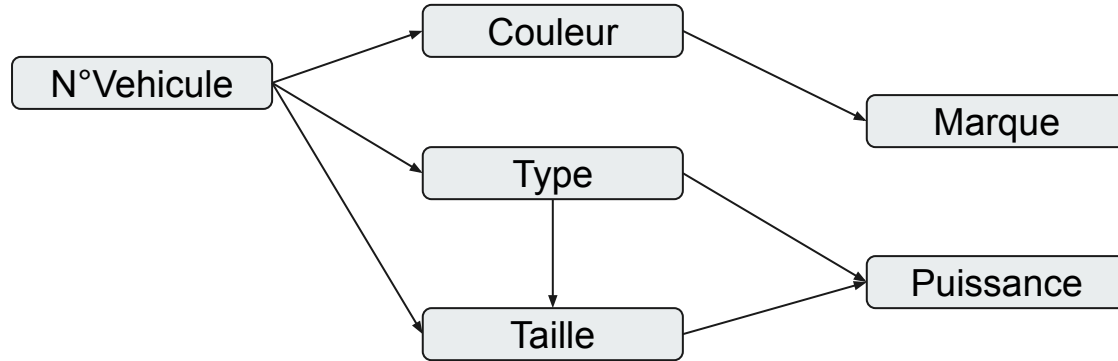
Fermeture Transitive

On appelle fermeture transitive d'un ensemble F de DFE, l'ensemble de toutes les DFE qui peuvent être composées par transitivité à partir des DFE de F



Fermeture Transitive

On appelle fermeture transitive d'un ensemble F de DFE, l'ensemble de toutes les DFE qui peuvent être composées par transitivité à partir des DFE de F



N°Vehicule → Couleur
N°Vehicule → Type
N°Vehicule → Taille
N°Vehicule → Marque
N°Vehicule → Puissance
Couleur → Marque
Type → Puissance
Type → Taille
Taille → Puissance



Les DF au service de la normalisation

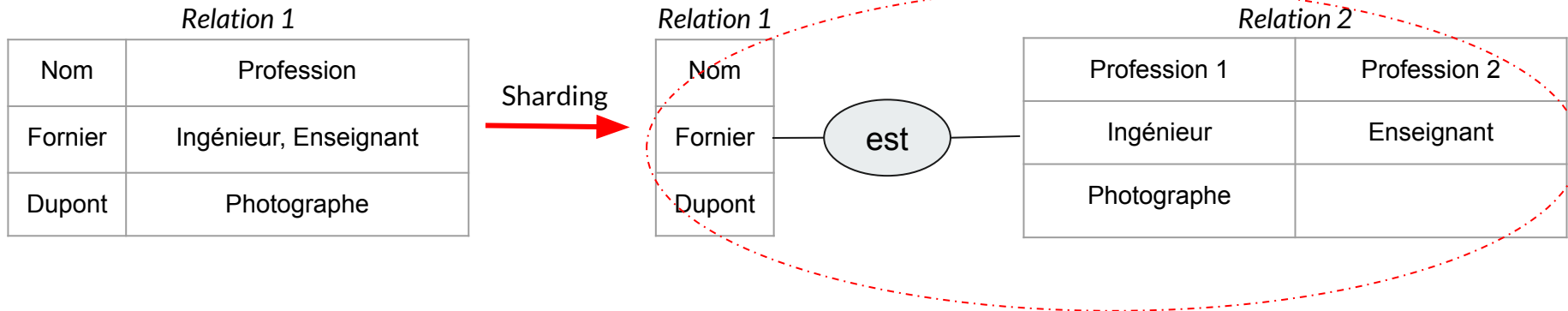
L'objectif de la normalisation est d'éviter d'observer des anomalies (d'écriture , de lecture...) provenant d'une mauvaise modélisation des données tout en préservant les DF et sans perdre d'informations.

Pour cela, on va définir des “formes normales” afin de pouvoir décomposer les tables pour éviter d'éventuelles anomalies et redondances.

La Première Forme Normale (ou NF1)

Une relation est dite en Première Forme Normale si et seulement si :

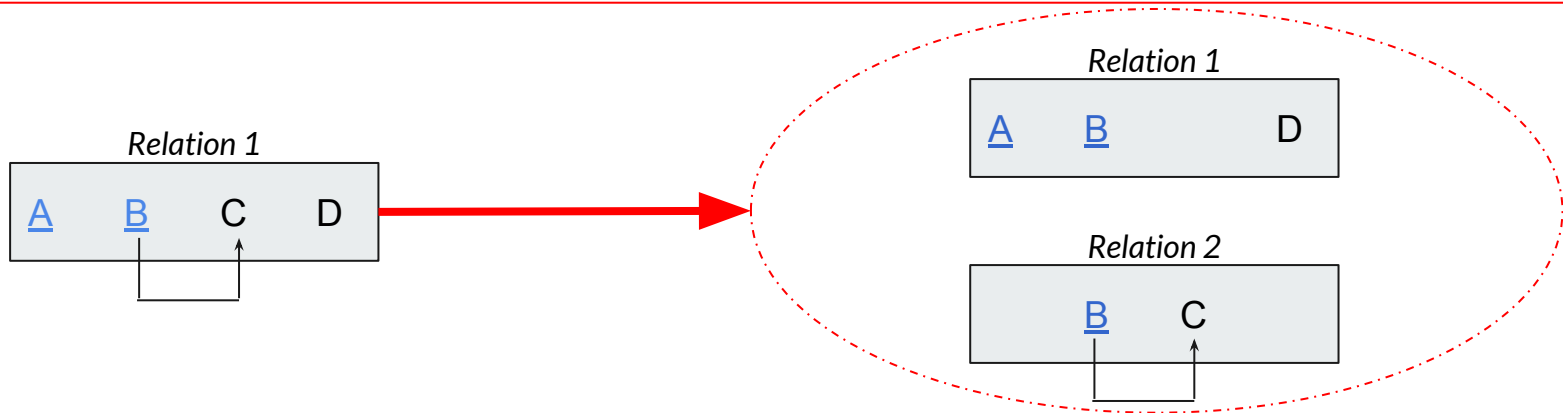
- **Tout attribut contient une valeur unique.**



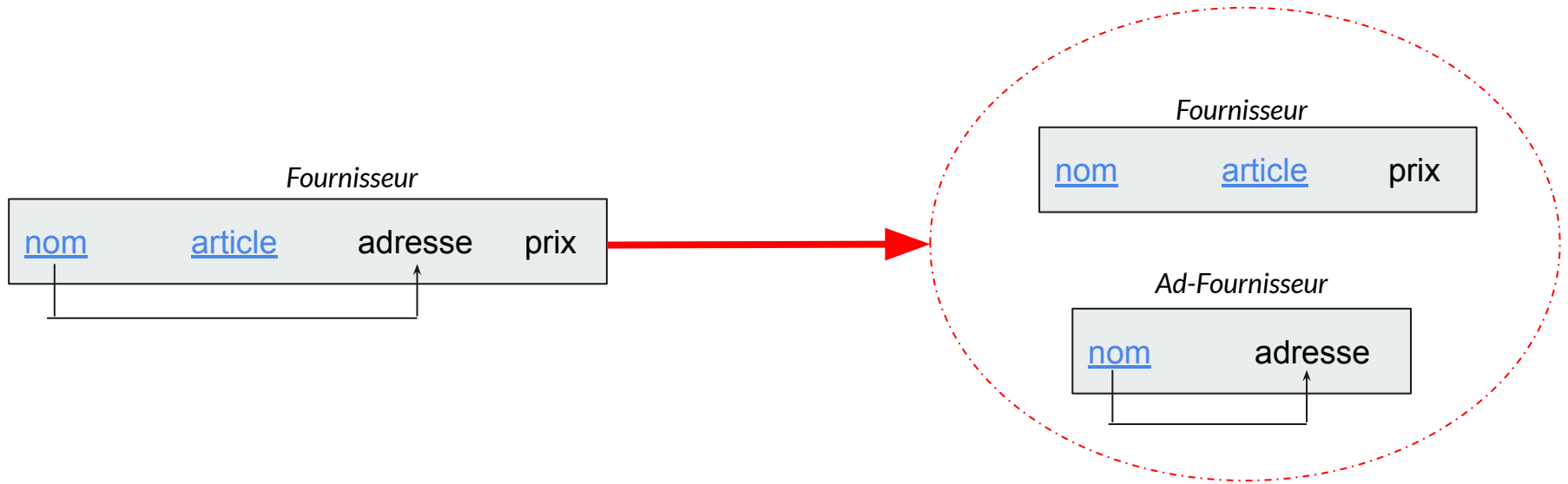
La Deuxième Forme Normale (ou NF2)

Une relation est dite en Deuxième Forme Normale si et seulement si :

- Elle est en NF1
- **Aucune partie de la clé ne détermine un autre attribut non clé.**



Exemple de NF2





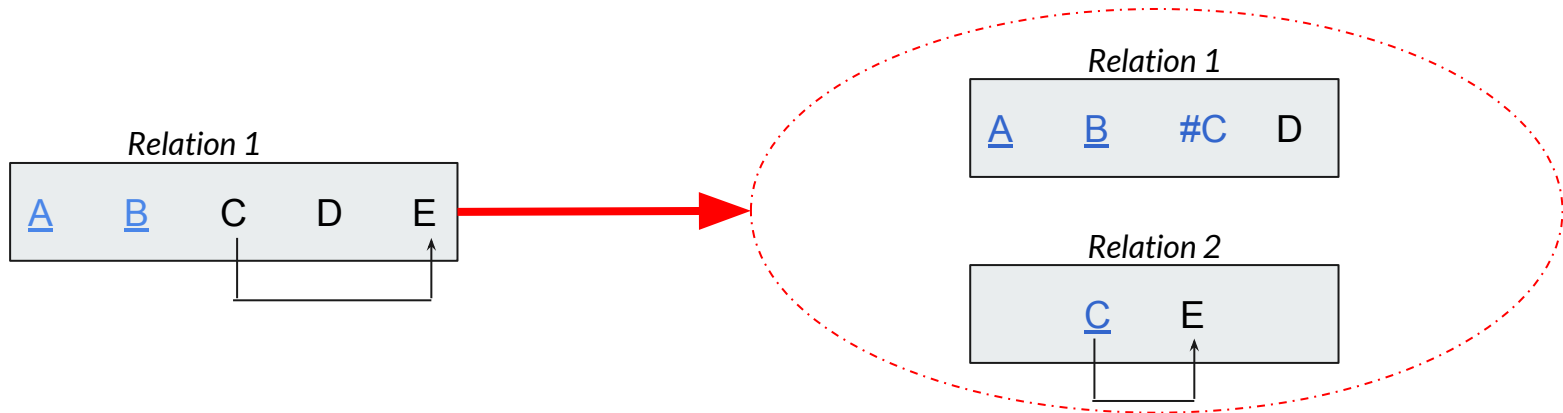
Propriété de la NF2

La deuxième forme normale permet d'éliminer les dépendances entre des parties de clé et des attributs n'appartenant pas à une clé.

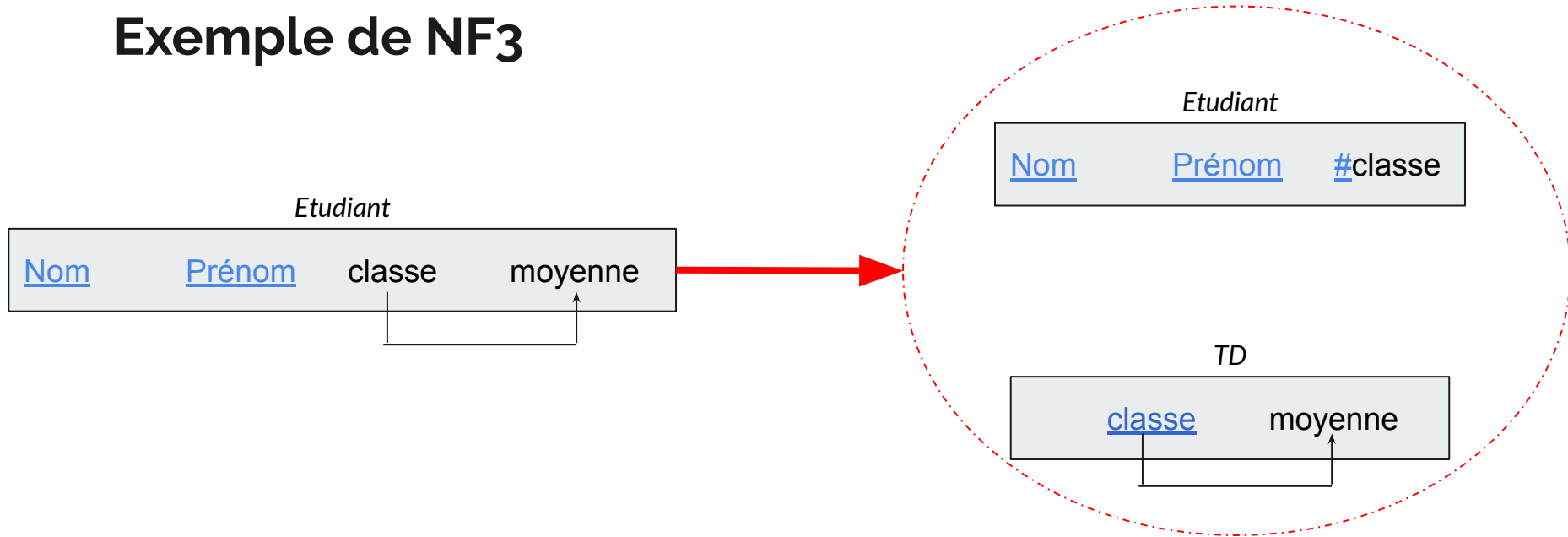
La Troisième Forme Normale (ou NF3)

Une relation est dite en Troisième Forme Normale si et seulement si :

- Elle est en 2NF
- **Il n'existe pas de dépendance entre les attributs non-clés.**



Exemple de NF3





Propriété de la NF3

- Il existe toujours une décomposition d'une relation R en relations R_1, R_2, \dots, R_n en NF3 telle qu'on ait ni perte de dépendances ni perte d'informations.
- Les dépendances fonctionnelles des relations décomposées permettent de revenir à celles de la relation initiale.
- Les relations décomposées permettent à tout instant de recomposer la relation initiale (notamment par jointures)
- Néanmoins, on crée de la redondance avec ce type de décomposition (création de clés étrangères)



Cours 4

TD2

Recherches de Dépendances Fonctionnelles Transformations en formes normales



TD/TP2

Rendez vous sur le fichier “TD TP BDDR.pdf” et commencez la section 2.



Cours 5

Présentation des Bases de données Relationnelles et non relationnelles

Présentation de l'écosystème

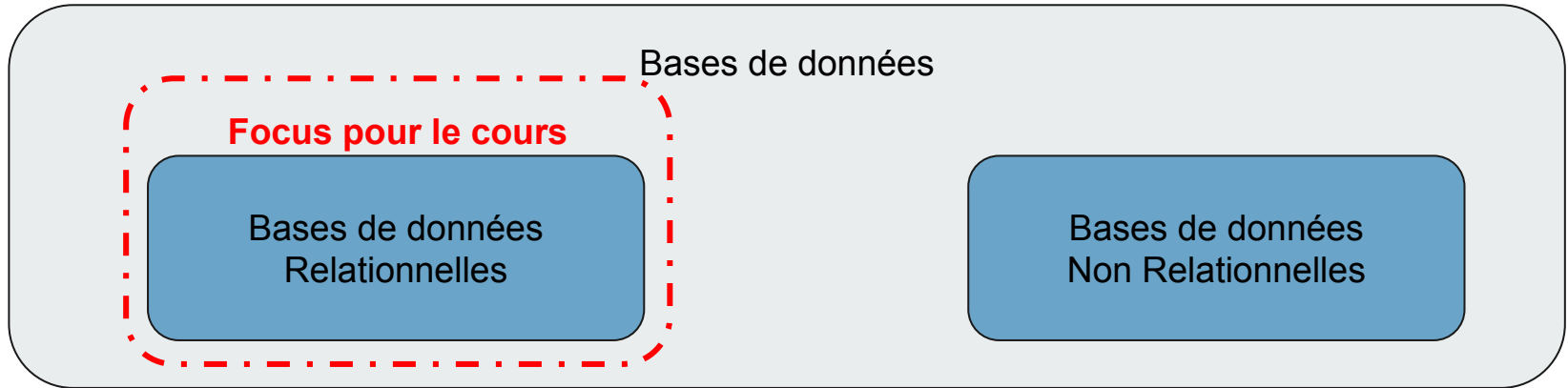
Installation de WampServer

Présentation d'une requête SQL - décomposition



Présentation des bases de données

Il existe aujourd'hui 2 grands types de bases de données : **Relationnelles** et **Non Relationnelles**.



Les technologies liées aux bases de données relationnelles



Bases de données
Relationnelles





MySQL

MySQL est un SGBDR (Système de Gestion de Bases de Données Relationnelles) open-source, développé par Oracle.

MySQL  SQL

Ne pas confondre MySQL qui est un SGBDR et SQL (Structured Query Language) qui est un langage qui permet d'effectuer des requêtes sur des bases de données relationnelles.



MySQL et MariaDB

MySQL et MariaDB sont des logiciels qui nous permettent d'exploiter une BDD. Ils servent d'interface entre l'utilisateur et la BDD.

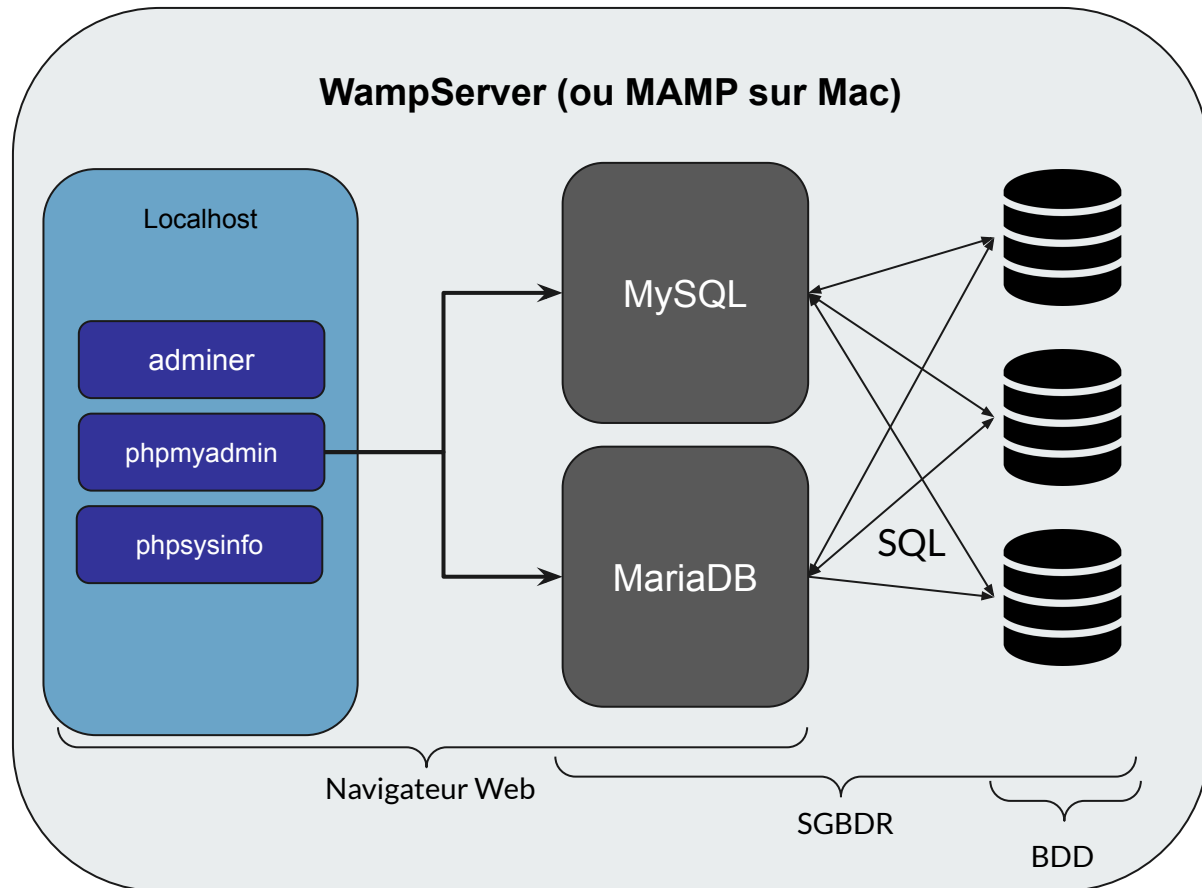




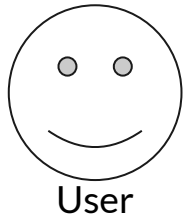
Installation de WampServer



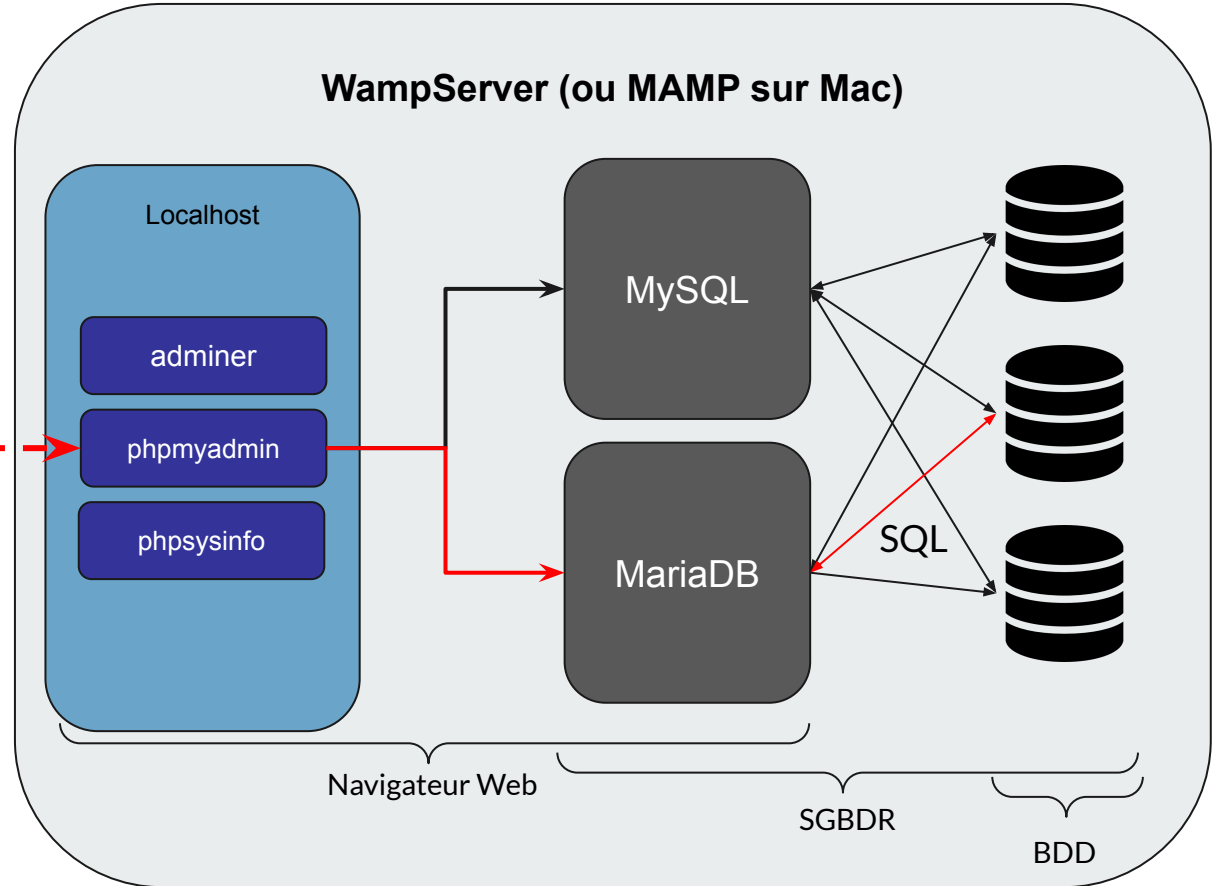
Architecture



Architecture



Se connecte



Installation de WAMP SERVER

Rendez-vous sur la page : <https://www.wampserver.com/>, téléchargez le logiciel et installez-le.



TÉLÉCHARGER WAMPSERVER 64 BITS (X64) 3.2.6

WampServer est disponible gratuitement (sous licence GPL). Vous pouvez remplir ce formulaire qui nous permettra de vous faire parvenir actualités formation d'Alter vway, société editrice, ainsi que toutes les informations liées aux évolutions de WampServer. Si vous ne le souhaitez pas, vous pourrez [passer au téléchargement direct](#).

Prénom :

Nom :

Société :

Email (*) :

Téléphone :

Pays :

Fonction (*) :

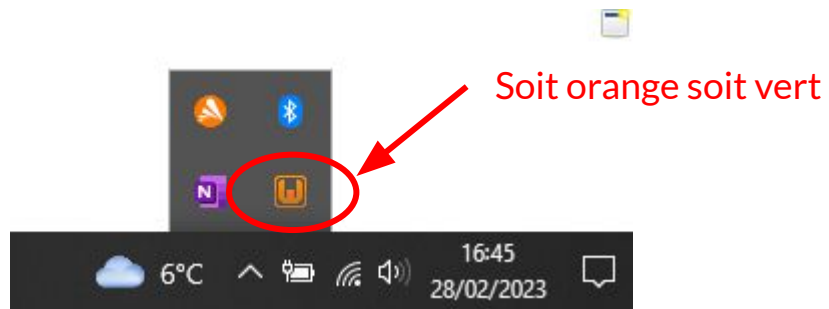
Vous avez des questions, des remarques, des commentaires ?

Votre utilisation de Wampserver :

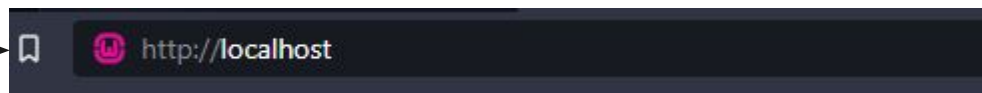
- ☐ Utilisation pour une application interne
- ☐ Utilisation pour développer en préproduction
- ☐ Ne prévoit pas d'utiliser WAMPSEVER

☐ Je souhaite recevoir des informations de WampServer

ENVOYER



Dans votre navigateur préféré





Wampserver

Apache 2.4 - MySQL 5 & 8 - MariaDB 10 - PHP 5 & 7

Version 3.2.3 - 64bit

french

classic

Configuration Serveur

Version Apache : 2.4.46 - [Documentation](#)

Server Software : Apache/2.4.46 (Win64) PHP/7.3.21 - Port défini pour Apache : 80

Version de PHP : 7.3.21 - [Documentation](#)

Extensions Chargées :

- apache2handler
- Core
- fileinfo
- hash
- ldap
- openssl
- Phar
- soap
- tokenizer
- xmlrpc
- zlib
- bcmath
- ctype
- filter
- iconv
- libxml
- pcre
- readline
- sockets
- wddx
- xmlwriter
- bz2
- date
- gd
- imap
- mbstring
- PDO
- Reflection
- SPL
- xdebug
- xsl
- calendar
- dom
- gettext
- intl
- mysqli
- pdo_mysql
- session
- sqlite3
- xml
- Zend OPcache
- com_dotnet
- exif
- gmp
- json
- mysqlnd
- pdo_sqlite
- SimpleXML
- standard
- xmlreader
- zip

Version de MySQL : 5.7.31 - Port défini pour MySQL : 3306 - SGBD par défaut - [Documentation MySQL](#)

Version de MariaDB : 10.4.13 - Port défini pour MariaDB : 3307 - [Documentation MariaDB](#) - [MySQL](#) - [MariaDB](#)

Outils

[phpinfo\(\)](#)

[phpmyadmin](#)

[Ajouter un Virtual Host](#)

Vos Projets

Aucun projet.
Pour en ajouter un nouveau, créez simplement un répertoire dans 'www'.

Vos Alias

- [adminer](#)
- [phpmyadmin](#)
- [phpsysinfo](#)

Vos VirtualHost

[localhost](#)



Bienvenue dans phpMyAdmin

Lange - *Language*

Français - French

Connexion ⓘ

Utilisateur :

Mot de passe :

Choix du serveur :

MySQL

Exécuter



Bienvenue dans phpMyAdmin

Lange - *Language*

Français - French

Connexion ⓘ

Utilisateur :

root

Mot de passe :

Choix du serveur :

MariaDB

Exécuter



Serveur courant :

MariaDB

Récentes Préférées



- Nouvelle base de données
- base1
- base2
- information_schema
- mysql
- performance_schema
- test
- test2

Paramètres généraux

Modifier le mot de passe

Interclassement pour la connexion au serveur : utf8mb4_unicode_ci

Plus de paramètres

Paramètres d'affichage

Langue - Language Français - French

Thème : pmahomme



Serveur courant :

MariaDB

Récentes Préférées

Nouvelle base de données

base1
base2
information_schema
mysql
performance_schema
test
test2

Paramètres généraux

Modifier le mot de passe

Interclassement pour la connexion au serveur : utf8mb4_unicode_ci



Plus de paramètres


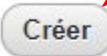
Paramètres d'affichage








Langue - Language Français - French

Thème : pmahomme

Bases de données

 Création d'une base de données 

| | Base de données ▲ | Interclassement | Action |
|--------------------------|--------------------|-------------------|---|
| <input type="checkbox"/> | base1 | latin1_swedish_ci |  Vérifier les privilèges |
| <input type="checkbox"/> | base2 | latin1_swedish_ci |  Vérifier les privilèges |
| <input type="checkbox"/> | information_schema | utf8_general_ci |  Vérifier les privilèges |
| <input type="checkbox"/> | mysql | latin1_swedish_ci |  Vérifier les privilèges |
| <input type="checkbox"/> | performance_schema | utf8_general_ci |  Vérifier les privilèges |
| <input type="checkbox"/> | test | latin1_swedish_ci |  Vérifier les privilèges |
| <input type="checkbox"/> | test2 | latin1_swedish_ci |  Vérifier les privilèges |
| Total : 7 | | | |

 ☐ Tout cocher Avec la sélection :  Supprimer



Serveur courant :

MariaDB

Récentes

Préférées

- Nouvelle base de données
- base1
- base2
- information_schema
- mysql
- performance_schema
- test
- test2
- tp_esgf

Serveur: MariaDB-3307 » Base de données: tp_esgf

Structure **SQL** Rechercher Requête Exporter

⚠ Aucune table n'a été trouvée dans cette base de données.

Nouvelle table

Nom: Nombre de colonnes:

phpMyAdmin

Serveur courant : MariaDB

Récentes Préférées

- Nouvelle base de données
- bdcommandes
- information_schema
- mysql
- performance_schema
- tp_esgf
 - Nouvelle table
 - commandes
 - fournisseurs
 - produits

Serveur: MariaDB:3307 » Base de données: tp_esgf

Structure SQL Rechercher Requête Exporter Importer Opérations Privileges Procédures stockées Événements Déclencheurs

Exécuter une ou des requêtes SQL sur la base de données « tp_esgf »:

```
1 DROP DATABASE IF EXISTS BDCOMMANDES; CREATE DATABASE BDCOMMANDES ;
2 USE BDCOMMANDES ;
3 DROP TABLE IF EXISTS Fournisseurs ; DROP TABLE IF EXISTS Produits ; DROP TABLE IF EXISTS Commandes ;
4 CREATE TABLE Fournisseurs (
5   fno Numeric(6) NOT NULL primary key ,
6   nom VARCHAR(25) NOT NULL ,
7   adresse VARCHAR(25) ,
8   ville VARCHAR(25) NOT NULL
9 ) ;
10
11 CREATE TABLE Produits (
12   pno Numeric(6) NOT NULL primary key ,
```

Copier/coller le script SQL

Effacer Format Récupérer la requête auto-sauvegardée

☐ Lier les paramètres

[Délimiteur ;] ☐ Afficher à nouveau la requête après exécution ☐ Conserver la boîte de requêtes ☐ ROLLBACK à la fin ☒ Activer la vérification des clés étrangères

Exécuter

<https://github.com/Le-Minh-Phuc/Relational-DB/blob/main/script1-db>



Serveur courant :

MariaDB

Récentes Préférées

990

- Nouvelle base de données
- bdcommandes
- information_schema
- mysql
- performance_schema
- tp_esgf
 - Nouvelle table
 - commandes
 - fournisseurs
 - produits

Serveur: MariaDB:3307 » Base de données: tp_esgf

Structure SQL Rechercher Requête Exporter Importer Opérations Privileges Procédures stockées

Afficher la zone SQL

✓ MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0011 seconde(s).)

DROP DATABASE IF EXISTS BDCOMMANDES

✓ MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0007 seconde(s).)

CREATE DATABASE BDCOMMANDES

✓ MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0003 seconde(s).)

USE BDCOMMANDES

✓ MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0009 seconde(s).)

DROP TABLE IF EXISTS Fournisseurs

✓ Affichage des lignes 0 - 9 (total de 10, traitement en 0,0003 seconde(s).)

```
SELECT * FROM `commandes`
```

☐ Tout afficher

Nombre de lignes : 25 ▼

Filtrer les lignes:

Chercher dans cette table

Trier pa

+ Options


| ← T → | | | | | | cno | fno | pno | qute | |
|--------------------------|---|--------|---|--------|---|-----------|------|-----|------|----|
| <input type="checkbox"/> |  | Éditer |  | Copier |  | Supprimer | 1001 | 17 | 103 | 10 |
| <input type="checkbox"/> |  | Éditer |  | Copier |  | Supprimer | 1003 | 15 | 103 | 2 |
| <input type="checkbox"/> |  | Éditer |  | Copier |  | Supprimer | 1005 | 17 | 102 | 1 |
| <input type="checkbox"/> |  | Éditer |  | Copier |  | Supprimer | 1007 | 15 | 108 | 1 |
| <input type="checkbox"/> |  | Éditer |  | Copier |  | Supprimer | 1011 | 19 | 107 | 12 |
| <input type="checkbox"/> |  | Éditer |  | Copier |  | Supprimer | 1013 | 13 | 107 | 5 |
| <input type="checkbox"/> |  | Éditer |  | Copier |  | Supprimer | 1017 | 19 | 105 | 3 |
| <input type="checkbox"/> |  | Éditer |  | Copier |  | Supprimer | 1019 | 14 | 103 | 10 |
| <input type="checkbox"/> |  | Éditer |  | Copier |  | Supprimer | 1023 | 10 | 102 | 8 |
| <input type="checkbox"/> |  | Éditer |  | Copier |  | Supprimer | 1029 | 17 | 108 | 15 |



☐ Tout cocher

Avec la sélection :

 Éditer

 Copier

 Supprimer

 Exporter

☐ Tout afficher

Nombre de lignes : 25 ▼

Filtrer les lignes:

Chercher dans cette table

Trier pa



Présentation d'une requête SQL

Une requête SQL pour une manipulation de donnée s'écrit d'une certaine manière :

[Action] [Element] FROM [Table] WHERE [Condition]



Exemples

“SELECT * FROM commandes” - Sélectionne toutes les entrées dans la table commandes

“SELECT * FROM commandes WHERE fno = 15” - Sélectionne toutes les entrées dont l’attribut “fno” est égal à 15.



A utilisier

<https://sql.sh/>



Cours 6

Les Manipulations de données



SELECT

L'utilisation la plus courante du SQL consiste à lire des données issues de la base de données. C'est ce qu'on appelle l'interrogation.

Son utilisation se fait de la manière suivante :

```
SELECT nom_du_champ FROM nom_du_tableau
```

***SELECT** * FROM commandes*



DISTINCT

DISTINCT se combine avec SELECT pour éviter des redondances dans les résultats il faut simplement ajouter DISTINCT après le mot SELECT.

“Sélectionner les différents fno dans la table commandes”

```
SELECT DISTINCT fno FROM commandes
```




WHERE

La commande **WHERE** dans une requête SQL permet d'extraire les lignes d'une base de données qui respectent une condition. Cela permet d'obtenir uniquement les informations désirées.

“Sélectionner les commandes dont le pno est supérieur à 104”

```
SELECT * FROM commandes WHERE pno > 104
```



AND & OR

Les opérateurs logiques AND et OR peuvent être utilisées dans la commande WHERE pour combiner des conditions.

Les opérateurs sont à ajoutés dans la condition WHERE. Ils peuvent être combinés à l'infini pour filtrer les données comme souhaités.

```
SELECT * FROM commandes WHERE fno>15 OR fno<12
```

```
SELECT * FROM commandes WHERE fno = 15 AND pno =12
```



IN

L'opérateur logique IN s'utilise avec la commande WHERE pour vérifier si une colonne est égale à une des valeurs comprise dans set de valeurs déterminés. C'est une méthode simple pour vérifier si une colonne est égale à une valeur OU une autre valeur OU une autre valeur et ainsi de suite, sans avoir à utiliser de multiple fois l'opérateur OR.



BETWEEN

L'opérateur BETWEEN est utilisé dans une requête SQL pour sélectionner un intervalle de données dans une requête utilisant WHERE. L'intervalle peut être constitué de chaînes de caractères, de nombres ou de dates. L'exemple le plus concret consiste par exemple à récupérer uniquement les enregistrements entre 2 dates définies.

```
SELECT * FROM commandes WHERE pno > 12 AND pno < 15
```

```
⇔ SELECT * FROM commandes WHERE pno BETWEEN 12 AND 15
```



LIKE

L'opérateur LIKE est utilisé dans la clause WHERE des requêtes SQL. Ce mot-clé permet d'effectuer une recherche sur un modèle particulier. Il est par exemple possible de rechercher les enregistrements dont la valeur d'une colonne commence par telle ou telle lettre. Les modèles de recherches sont multiples.

- **LIKE '%a'** : Le caractère “%” est un caractère joker qui remplace tous les autres caractères. Ainsi, ce modèle **permet de rechercher toutes les chaînes de caractères qui se terminent par un “a”**.
- **LIKE 'a%'** : Ce modèle permet de rechercher toutes les chaînes de caractères qui **commencent par un “a”**.
- **LIKE '%a%'** : Ce modèle est utilisé pour rechercher **tous les enregistrements qui utilisent le caractère “a”**.
- **LIKE 'pa%on'** : Ce modèle permet de rechercher les chaînes qui **commencent par “pa” et qui se terminent par “on”** (“pantalon”, “pardon”...)
- **LIKE 'a_c'** : peu utilisé, le caractère “_” (underscore) **peut être remplacé par n'importe quel caractère, mais un seul caractère uniquement** (alors que le symbole pourcentage “%” peut être remplacé par un nombre incalculable de caractères). Ainsi, ce modèle permet de retourner les lignes “aac”, “abc” ou même “azc”.



IS NULL / IS NOT NULL

L'opérateur IS permet de filtrer les résultats qui contiennent la valeur NULL. Cet opérateur est indispensable car la valeur NULL est une valeur inconnue et ne peut par conséquent pas être filtrée par les opérateurs de comparaison (cf. égal, inférieur, supérieur ou différent).



Cours 7

Les Créations et modifications de table



CREATE DATABASE / CREATE TABLE

La création d'une base de données en SQL est possible en ligne de commande grâce à la commande CREATE DATABASE. Même si les systèmes de gestion de base de données (SGBD) sont souvent utilisés pour créer une base, il convient de connaître la commande à utiliser, qui est très simple. Pour créer une table dans une base de données, on utilise CREATE TABLE.

```
CREATE DATABASE IF NOT EXISTS esgf_db
```

```
CREATE TABLE IF NOT EXISTS commandes
```




ALTER TABLE

La commande ALTER TABLE permet de modifier une table existante. Idéal pour ajouter une colonne, supprimer une colonne ou modifier une colonne existante, par exemple pour changer le type.

ALTER TABLE commandes

ADD nom_commande CHAR

```
ALTER TABLE nom_table
```

```
ADD nom_colonne type_donnees
```



UPDATE

La commande UPDATE permet d'effectuer des modifications sur des lignes existantes. Très souvent cette commande est utilisée avec WHERE pour spécifier sur quelles lignes doivent porter la ou les modifications.

```
UPDATE table
```

```
SET nom_colonne_1 = 'nouvelle valeur'
```

```
WHERE condition
```



DELETE

La commande DELETE en SQL permet de supprimer des lignes dans une table. En utilisant cette commande associé à WHERE il est possible de sélectionner les lignes concernées qui seront supprimées.

```
DELETE FROM `table`
```

```
WHERE condition
```



TRUNCATE TABLE

En SQL, la commande **TRUNCATE TABLE** permet de supprimer toutes les données d'une table sans supprimer la table en elle-même. En d'autres mots, cela permet de purger la table. Cette instruction diffère de la commande **DROP** qui à pour but de supprimer les données ainsi que la table qui les contient.



GROUP BY

La commande GROUP BY est utilisée en SQL pour grouper plusieurs résultats et utiliser une fonction de totaux sur un groupe de résultat. Sur une table qui contient toutes les ventes d'un magasin, il est par exemple possible de liste regrouper les ventes par clients identiques et d'obtenir le coût total des achats pour chaque client.

```
SELECT colonne1, fonction(colonne2)
```

```
FROM table
```

```
GROUP BY colonne1
```



ORDER BY

La commande ORDER BY permet de trier les lignes dans un résultat d'une requête SQL. Il est possible de trier les données sur une ou plusieurs colonnes, par ordre ascendant ou descendant.

```
SELECT colonne1, colonne2
```

```
FROM table
```

```
ORDER BY colonne1
```



AS

Dans le langage SQL il est possible d'utiliser des **alias** pour renommer temporairement une colonne ou une table dans une requête. Cette astuce est particulièrement utile pour faciliter la lecture des requêtes.

```
SELECT * FROM commandes AS com WHERE com.pno = 12
```



Cours 8

Requêtes SQL



TD/TP3

Rendez vous sur le fichier “TD TP BDDR.pdf” et commencez la section 3 - SQL1



Cours 9

Requêtes SQL



TD/TP3

Rendez vous sur le fichier “TD TP BDDR.pdf” et continuez la section 3 - SQL1



Cours 10

Les fonctions d'agrégation



HAVING

La condition HAVING en SQL est presque similaire à WHERE à la seule différence que HAVING permet de filtrer en utilisant des fonctions telles que SUM(), COUNT(), AVG(), MIN() ou MAX().

```
SELECT colonne1, SUM(colonne2)
```

```
FROM nom_table
```

```
GROUP BY colonne1
```

```
    HAVING fonction(colonne2) operateur valeur
```



AVG

La fonction d'agrégation `AVG()` dans le langage SQL permet de calculer une valeur moyenne sur un ensemble d'enregistrement de type numérique et non nul.



COUNT

En SQL, la fonction d'agrégation COUNT() permet de compter le nombre d'enregistrement dans une table. Connaître le nombre de lignes dans une table est très pratique dans de nombreux cas, par exemple pour savoir combien d'utilisateurs sont présents dans une table ou pour connaître le nombre de commentaires sur un article.



MAX

Dans le langage SQL, la fonction d'agrégation MAX() permet de retourner la valeur maximale d'une colonne dans un set d'enregistrement. La fonction peut s'appliquer à des données numériques ou alphanumériques. Il est par exemple possible de rechercher le produit le plus cher dans une table d'une boutique en ligne.



MIN

La fonction d'agrégation MIN() de SQL permet de retourner la plus petite valeur d'une colonne sélectionnée. Cette fonction s'applique aussi bien à des données numériques qu'à des données alphanumériques.



SUM

Dans le langage SQL, la fonction d'agrégation SUM() permet de calculer la somme totale d'une colonne contenant des valeurs numériques. Cette fonction ne fonctionne que sur des colonnes de types numériques (INT, FLOAT ...) et n'additionne pas les valeurs NULL.



Cours 11

Requêtes SQL



TD/TP3

Rendez vous sur le fichier “TD TP BDDR.pdf” et commencez la section 3 - SQL2



Cours 12

Évaluation sur table



Cours 13

Jointures



INNER JOIN

Dans le langage SQL la commande INNER JOIN, aussi appelée EQUIJOIN, est un type de jointures très communes pour lier plusieurs tables entre-elles. Cette commande retourne les enregistrements lorsqu'il y a au moins une ligne dans chaque colonne qui correspond à la condition.

```
SELECT *
```

```
FROM table1
```

```
INNER JOIN table2 ON table1.id = table2.fk_id
```



CROSS JOIN

Dans le langage SQL, la commande CROSS JOIN est un type de jointure sur 2 tables SQL qui permet de retourner le produit cartésien. Autrement dit, cela permet de retourner chaque ligne d'une table avec chaque ligne d'une autre table. Ainsi effectuer le produit cartésien d'une table A qui contient 30 résultats avec une table B de 40 résultats va produire 1200 résultats ($30 \times 40 = 1200$). En général la commande CROSS JOIN est combinée avec la commande WHERE pour filtrer les résultats qui respectent certaines conditions.

```
SELECT *  
  
FROM table1  
  
    CROSS JOIN table2
```




LEFT JOIN

Dans le langage SQL, la commande LEFT JOIN (aussi appelée LEFT OUTER JOIN) est un type de jointure entre 2 tables. Cela permet de lister tous les résultats de la table de gauche (left = gauche) même s'il n'y a pas de correspondance dans la deuxième tables.

```
SELECT *
```

```
FROM table1
```

```
LEFT JOIN table2 ON table1.id = table2.fk_id
```



RIGHT JOIN

En SQL, la commande RIGHT JOIN (ou RIGHT OUTER JOIN) est un type de jointure entre 2 tables qui permet de retourner tous les enregistrements de la table de droite (right = droite) même s'il n'y a pas de correspondance avec la table de gauche. S'il y a un enregistrement de la table de droite qui ne trouve pas de correspondance dans la table de gauche, alors les colonnes de la table de gauche auront NULL pour valeur.

```
SELECT *
```

```
FROM table1
```

```
RIGHT JOIN table2 ON table1.id = table2.fk_id
```



FULL JOIN

Dans le langage SQL, la commande FULL JOIN (ou FULL OUTER JOIN) permet de faire une jointure entre 2 tables. L'utilisation de cette commande permet de combiner les résultats des 2 tables, les associer entre eux grâce à une condition et remplir avec des valeurs NULL si la condition n'est pas respectée.

```
SELECT *  
  
FROM table1  
  
FULL JOIN table2 ON table1.id = table2.fk_id
```



SELF JOIN

En SQL, un SELF JOIN correspond à une jointure d'une table avec elle-même. Ce type de requête n'est pas si commun mais très pratique dans le cas où une table lie des informations avec des enregistrements de la même table.

```
SELECT `t1`.`nom_colonne1`, `t1`.`nom_colonne2`, `t2`.`nom_colonne1`, `t2`.`nom_colonne2`  
  
FROM `table` as `t1`  
  
LEFT OUTER JOIN `table` as `t2` ON `t2`.`fk_id` = `t1`.`id`
```



NATURAL JOIN

Dans le langage SQL, la commande NATURAL JOIN permet de faire une jointure naturelle entre 2 tables. Cette jointure s'effectue à la condition qu'il y ai des colonnes du même nom et de même type dans les 2 tables. Le résultat d'une jointure naturelle est la création d'un tableau avec autant de lignes qu'il y a de paires correspondant à l'association des colonnes de même nom.

```
SELECT *
```

```
FROM table1
```

```
NATURAL JOIN table2
```



Cours 14

Requêtes SQL



TD/TP3

Rendez vous sur le fichier “TD TP BDDR.pdf” et continuez la section 3 - SQL2 - Jointures.