Cours – Bases de données relationnelles

Objectifs :

- ⇨ Découvrir le concept de base de données
- ⇨ Présenter le rôle et l'intérêt des SGBD
- ⇨ Comprendre le modèle relationnel
- ⇨ Apprendre à créer une base de données relationnelle

I – Qu'est-ce qu'une base de données ?

1) Historique

Avec les premiers ordinateurs les systèmes de stockage étaient des cartes perforées (pour les programmes ou de petites quantités de données) et les bandes magnétiques (pour les grandes quantités de données).
Lorsqu'ils devaient manipuler beaucoup de données, les ordinateurs devaient donc lire une partie de la bande magnétique et la mettre en mémoire, traiter les données, puis retransférer le résultat sur bande avant de passer à la partie suivante.
L'invention du **disque dur** par IBM en 1956, a permis d'accéder *directement* à de grandes quantités d'informations[1]. Il n'y a alors plus la nécessité de traiter les données par lots comme avec les bandes magnétiques et un programme peut ainsi accéder potentiellement à l'ensemble des données.
Le terme *database* (base de données) est apparu en 1964 pour désigner une collection d'informations partagées par différents utilisateurs d'un système d'informations militaire.
En 1970 le Britannique **Edgard J. Codd** théorise le **modèle relationnel** qui permet de traiter de façon efficace des données en grand nombre. Ce modèle, qui est le seul au programme de NSI, est à ce jour le modèle de base de données le plus utilisé. Avec le Big Data[2] et des usages spécifiques comme les réseaux sociaux ou l'intelligence artificielle, on observe cependant la montée en puissance d'autres modèles non relationnels.

2) Les SGBD

Avec la généralisation du traitement de volumes de données très importants, il a fallu développer de véritables **S**ystèmes de **G**estion de **B**ase de **D**onnées (abrégé SGBD en français ou *DBMS* pour ***D**ataBase **M**anagement **S**ystem* en anglais).
Ces logiciels spécialisés ont pour but de gérer une ou plusieurs bases de données. Ils permettent notamment de :

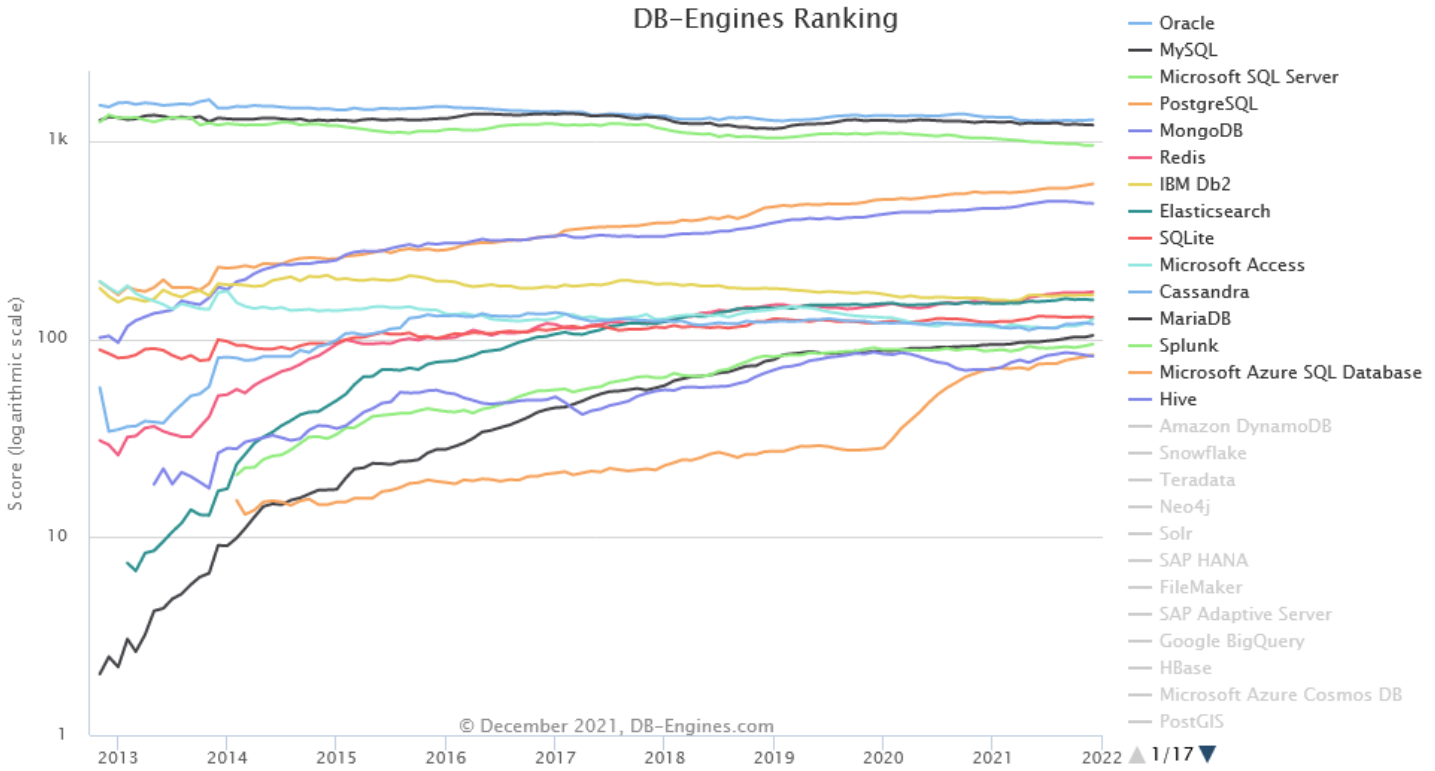
- gérer la **lecture, l'écriture ou la modification** des informations contenues dans une base de données ;
- gérer les **autorisations d'accès** à la base de données. Il est en effet souvent nécessaire de contrôler les accès par exemple en permettant à l'utilisateur A de lire et d'écrire dans la base de données alors que l'utilisateur B aura uniquement la possibilité de lire les informations contenues dans cette même base de données et que l'utilisateur C ne pourra travailler que sur une partie spécifique de la base de données.
- Gérer le **stockage des fichiers** de la base de données. Ces fichiers doivent pouvoir être sauvegardés sur plusieurs supports y compris lorsque la base de données est en fonctionnement.
- Gérer la **redondance des données**. Le système maintient plusieurs copies des données pour être capable de continuer à fonctionner si un des systèmes de stockage a une défaillance. Le SGBD se charge de créer

[1] Le premier disque dur crée par IBM pouvait stocker 5 Mo pour un coût d'environ 50 000 $. Depuis les capacités n'ont cessé d'augmenter et les coûts de diminuer.
[2] Avec l'apparition du Web, la quantité de données à stocker a littéralement explosé. Le « Big Data » désigne ainsi les immenses quantités de données qui sont collectées et disponibles sur le web. Ces données sont tellement volumineuses qu'elles ne peuvent tenir dans la mémoire d'un seul ordinateur.

et de synchroniser les copies ainsi que le basculement d'un système de stockage à l'autre en fonction de l'état de fonctionnement et pour répartir la charge.

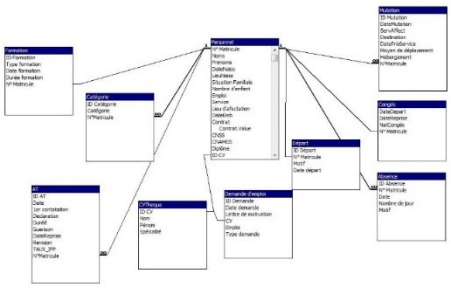
- Gérer les **accès concurrents** à la base de données. Plusieurs utilisateurs doivent pouvoir accéder simultanément à la base de données. C'est le SGBD qui gère les problèmes, notamment si 2 personnes désirent modifier la même donnée au même moment.
- **Effectuer des transactions** de bout en bout. Une transaction est une série d'opérations qui doivent être effectuées toutes de bout en bout ou bien pas du tout (c'est du tout ou rien). C'est ce qui se passe par exemple lorsqu'on utilise un moyen de paiement en ligne.

Il existe aujourd'hui de très nombreux SGBD, les principaux étant Oracle, MySQL (open source appartenant à Oracle) et Microsoft SQL server.



https://db-engines.com/en/ranking

3) Le modèle relationnel



Le modèle tabulaire vu en première NSI (voir le chapitre sur les données en table et fichiers csv) est adapté pour de petites quantités de données qui ne soient pas trop complexes mais se révèle insuffisant lorsque la quantité de données est trop importante[3] ou trop complexe (et nécessiterai un nombre de colonnes trop important par exemple).
Le modèle relationnel permet de stocker les données sous la forme de plusieurs tables (les relations) liées entre elles et possédant certaines contraintes permettant de les gérer efficacement.
Il s'appuie sur des théories mathématiques portant sur les ensembles qui permettent d'optimiser et de rationaliser les opérations.
Les SGBD qui sont fondés sur le modèle relationnel sont dit « SGBD relationnel » ou SGBRD (*RDBMS* avec R pour *Relational* en anglais).

[3] Voir le problème dans la gestion du covid 19 crée par les limitations d'Excel sur https://www.zdnet.fr/actualites/au-royaume-uni-16-000-malades-atteints-de-la-covid-19-oublies-sur-un-fichier-excel-39910779.htm.

II - Base de données relationnelle

Nous allons aborder cette partie en prenant l'exemple d'un système un peu comme pronote, permettant de gérer les notes des élèves d'un lycée.

1) Limites du modèle tabulaire

Admettons que nous voulions gérer les notes des élèves avec un grand fichier csv. Celui-ci devrait contenir de nombreuses colonnes comme par exemple :

	Nom	Prénom	Classe	Groupe	Date	Matière	Service	Note	sur	Coef.	Professeur	Intitulé
1	Zeblouse	Agathe	TG3	1	12/12/2021	Math	Ecrit	8,5	20	2	Tournesol	DS Géométrie
2	Desartites	André	1G5	2	08/10/2021	Espagnol	Oral	7	10	0,5	Ramirez	Compréhension
3	Zetaufré	Mélanie	2GT6	1	16/10/2021	Physique		17	20	3	Allister	Exposé
4	Fémal	Aïcha	2GT3	1	22/09/2021	SNT	Ecrit	12	15	1	Hector	Interro #2
5	Legnidu	Clotaire	2GT3	1	22/09/2021	SNT	Ecrit	4	15	1	Hector	Interro #2
6	Coppi	Fausto	2GT3	1	22/09/2021	SNT	Ecrit	11,5	15	1	Hector	Interro #2
7	Lovelace	Ada	1G2	2	24/11/2021	NSI		19	20	2	Babbage	TP langages

Cette représentation présente de nombreux problèmes comme par exemple :

- Impossibilité d'insérer un nouvel élève sans lui mettre de note
- Deux professeurs qui ont le même nom ne pourront pas être distingués
- Cette table ne permet pas par exemple de gérer également les absences des élèves
- Redondance inutile de certaines informations (comme l'interro #2 de SNT qui apparaît sur les lignes 4 à 6).

Pour résoudre en grande partie ces problèmes, il nous faut une meilleure organisation des données. C'est ce que permet le modèle relationnel.

2) Structure d'une base de données relationnelle

Dans le modèle relationnel, les données sont stockées dans des tables qui correspondent à des **relations**. Chaque relation va porter sur un des aspects des données.

a. Relations

Voyons par exemple la relation `eleve` :

On remarque que le nom de la relation et ceux des attributs ne possèdent pas d'accents, sont écrits en minuscule et que la relation est écrite au singulier (`eleve` et non pas `eleves`). Ce sont de bonnes pratiques qui permettent d'éviter les problèmes sur les noms mais pas une obligation.

nom	prenom	classe	naissance	entree	sortie
Zeblouse	Agathe	TG3	03/11/04	03/09/21	
Desartites	André	1G5	27/08/05	03/09/21	22/10/21
Zetaufré	Mélanie	2GT6	06/04/06	03/09/21	
Fémal	Aïcha	2GT3	13/09/06	17/09/21	
Legnidu	Clotaire	2GT3	25/02/05	03/09/21	
Coppi	Fausto	2GT3	14/03/06	03/09/21	18/12/21
Lovelace	Ada	1G2	20/06/05	03/09/21	
Dubois	Alix	TG1	05/01/04	03/09/21	

En-tête

Enregistrement ou **n-uplet** ou **tuple**

attribut

Table ou **relation**

On appelle **ordre** d'une relation le nombre de ses attributs.

Ici la relation `eleve` est d'ordre 6.

On appelle **cardinal** d'une relation le nombre de ses n-uplets.

Ici la relation `eleve` a un cardinal de 8.

b. Contraintes d'intégrité

De manière à garder un maximum de cohérence à la base de données, on impose un certain nombre de contraintes sur les données de la table.

- Contrainte de domaine

On comprend bien que donner la valeur « abc » à l'attribut « naissance » risque de poser problème. Pour limiter les problèmes de cohérence des données et simplifier leur stockage, on va définir un **domaine** pour chaque attribut.
Ces domaines correspondent un peu aux types de donnée en python.
Il existe différents type « standards » et chaque SGBD peut en proposer des variantes ou d'autres types (par exemple le type booléen).
La table ci-contre présente les principaux domaines standard proposés par tous les SGBD.
Les types entiers peuvent être signés ou non signés (que des nombres positifs – dans ce cas on rajoute le mot-clé UNSIGNED dans le nom de domaine).

Nom	Exact/ approché	Description
SMALLINT	Exact	Nombre entier sur 16 bits
INTEGER, INT	Exact	Nombre entier sur 32 bits
BIGINT	Exact	Nombre entier sur 64 bits
DECIMAL(X,Y), NUMERIC(X,Y)	Exact	Nombre décimal de X chiffres, dont Y après la virgule
FLOAT(X), REAL	Approché	Flottant dont la mantisse est sur X bits. REAL est un alias pour FLOAT(24)
CHAR(n)		Texte d'exactement n caractères
VARCHAR(n)		Texte d'au plus n caractères
TEXT		Texte de taille quelconque
DATE		Date au format AAAA-MM-JJ
TIME		Heure au format HH:MM:SS
TIMESTAMP		Date et heure au format AAAA-MM-JJ HH:MM:SS

Application 1 :
Proposer des domaines pour chaque attribut de la relation `eleve`.

Remarque importante : Valeurs NULL et logique ternaire

Quel que soit son domaine, un attribut peut ne pas avoir de valeur. Dans ce cas, le SGBD lui attribue la valeur spéciale NULL.

L'autorisation de valeurs NULL dans les attributs introduit une logique ternaire dans le SGBD. Une comparaison peut alors donner une des trois conditions : True, False et Unknown

Étant donné que la valeur NULL est considérée comme inconnue, deux valeurs NULL comparées l'une avec l'autre ne sont pas considérées comme égales. Dans les expressions utilisant des opérateurs arithmétiques, si l'un des opérandes a la valeur NULL, le résultat est également NULL.

De même les comparaisons à une valeur NULL vont donner un résultat Unkown qui n'est pas vrai (et donc la comparaison va se comporter comme si le résultat était False).

& (AND)	T	U	F
T	T	U	F
U	U	U	F
F	F	F	F

I (OR)	T	U	F
T	T	T	T
U	T	U	U
F	T	U	F

~(NOT)	
T	F
U	U
F	T

Tables de vérité des opérateurs logiques
T : True, U : Unknown, F : False

On peut forcer un attribut à ne pas accepter la valeur NULL (en précisant `NOT NULL` dans le schéma relationnel (voir plus loin)) et/ou lui attribuer une valeur par défaut si aucune valeur n'est précisée (avec `DEFAULT`).

- Contrainte d'entité (ou de relation)

La contrainte d'entité permet d'assurer que tout enregistrement soit unique : cette contrainte est réalisée par l'existence obligatoire d'une **clé primaire** (*primary key* en anglais).

Ainsi pour la relation `eleve`, un n-uplet peut être identifié de manière unique par le nom, le prénom et la classe. Le triplet (*nom*, *prenom*, *classe*) est donc une clé primaire de la relation `eleve`.

Une **clé primaire** est un attribut ou un ensemble d'attribut d'une relation qui permet d'identifier de manière unique un enregistrement de la relation.

Si une autre relation doit faire référence à un enregistrement de la table `eleve`, il n'est cependant pas pratique de devoir utiliser à chaque fois ce triplet. De plus il est possible (mais fortement improbable) que deux élèves ayant le même nom et le même prénom soient dans la même classe. Pour éviter ces écueils, on crée souvent de manière artificielle une clé primaire en utilisant un **numéro d'identifiant unique**.

Voici ce que pourrait donner la relation `eleve` avec cet identifiant (nommé ici « *id* ») :

id	nom	prenom	classe	naissance	entree	sortie
1	Zeblouse	Agathe	TG3	03/11/04	03/09/21	
2	Desartites	André	1G5	27/08/05	03/09/21	22/10/21
3	Zetaufré	Mélanie	2GT6	06/04/06	03/09/21	
4	Fémal	Aïcha	2GT3	13/09/06	17/09/21	
5	Legnidu	Clotaire	2GT3	25/02/05	03/09/21	
6	Coppi	Fausto	2GT3	14/03/06	03/09/21	18/12/21
7	Lovelace	Ada	1G2	20/06/05	03/09/21	
8	Dubois	Alix	TG1	05/01/04	03/09/21	

Il peut exister plusieurs clés permettant d'identifier un enregistrement de façon unique dans la relation. De telles clés sont appelées **clé candidates**. La clé primaire est donc une clé candidate particulière.

- Contrainte de référence

Dans notre système nous souhaitons pouvoir enregistrer les notes des élèves. On va donc créer une deuxième relation : `note`[4] (voir ci-contre).
Cette relation doit référencer les élèves qui se trouvent dans la table `eleve` ainsi que les évaluations qui se trouvent dans la table `evaluation`.
Il faut bien sûr s'assurer que les n-uplets désignés par *id_eleve* dans la relation note existent bien dans la relation eleve.
Pour ce faire, il faut indiquer au SGBD que *id_eleve* et *id_eval* sont des **clés étrangères** (*foreign key* en anglais) qui référencent respectivement *id* dans la relation `eleve` et *id* dans la relation `evaluation`.

id_eleve	id_eval	note	symbole
12	5	12,5	
153	38	0	N.Rendu
4	23	12	
5	23	4	
6	23	11,5	
41	23		Abs
19	65	0	Abs
87	122	38	

Relation note

Une **clé étrangère** est un attribut d'une relation qui doit être égal à une clé primaire d'une autre relation (on dit que la clé étrangère *référence* cet attribut de l'autre table).

- Contraintes utilisateurs (ou contraintes de métier)

On place dans cette catégorie de contrainte toutes les contraintes que l'on ne peut pas exprimer avec les trois catégories précédentes. Certaines contraintes pourront être gérées directement par le SGBD mais des contraintes plus complexes doivent parfois être mises en place dans le logiciel qui communique avec le SGBD.
Par exemple sur la relation `eleve`, on pourrait donner comme contrainte utilisateur que l'attribut *sortie* s'il existe doit toujours être postérieur ou égal à l'attribut *entree*.

Application 2 :
Proposer d'autres contraintes utilisateur sur les relations `eleve` ou `evaluation`.

c. Schéma relationnel

Le schéma relationnel est l'ensemble des relations présentes dans une base de données.

[4] Notez toujours l'emploi du singulier (« note » et non « notes »).

Le schéma relationnel d'une base de données contient les informations suivantes :

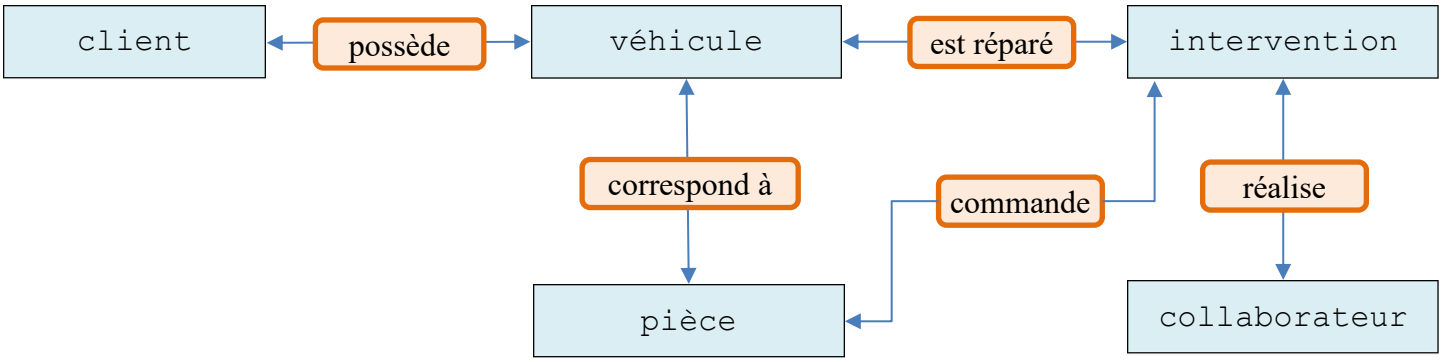
- les noms des différentes relations ;
- pour chaque relation, la liste des attributs avec leur domaine respectif ;
- pour chaque relation, la clé primaire et éventuellement les clés étrangères.

La syntaxe habituellement utilisée est la suivante :
`Nom_de_la_relation` (*attribut1* domaine, *attribut2* domaine, …)
Le ou les attribut(s) constituant la clé primaire est(sont) souligné(s) et les clés étrangères sont soulignées en pointillés (ou suivies d'un signe « # »).
Pour la base de données que nous avons prise en exemple depuis le début, cela donnerait :
`eleve` (*id* INTEGER UNSIGNED, *nom* VARCHAR(60), *prenom* VARCHAR(60),
classe VARCHAR(10), *naissance* DATE, *entree* DATE, *sortie* DATE)
`note` (*id_eleve* INTEGER UNSIGNED, *id_eval* INTEGER UNSIGNED, *note* DECIMAL(5,2),
symbole VARCHAR(12))

Application 3 :
1) Proposer un schéma relationnel pour la relation `evaluation`.
2) Donner un schéma relationnel pour la ou les relations permettant de gérer les absences des élèves.

3) Conception et représentation d'une base de données

Pour concevoir une base de données, on utilise généralement le modèle entité-association.
Imaginons par exemple que l'on souhaite écrire une base de données pour un garage qui répare et entretien des véhicules.
La base doit pouvoir contenir les clients ainsi que les véhicules. Chaque **client** peut *posséder* un ou plusieurs **véhicules**. Le garage emploie des **collaborateurs** qui peuvent *réaliser* des **interventions** sur les véhicules. A chaque intervention sur un véhicule, on peut être amené à *commander* des **pièces** qui *correspondent* au véhicule réparé ou entretenu.
Dans l'exemple précédent, les mots en gras représentent les entités de la base et ceux en italique (les verbes d'action) représentent les associations. On peut alors réaliser un schéma entité-association pour la base de données garage :



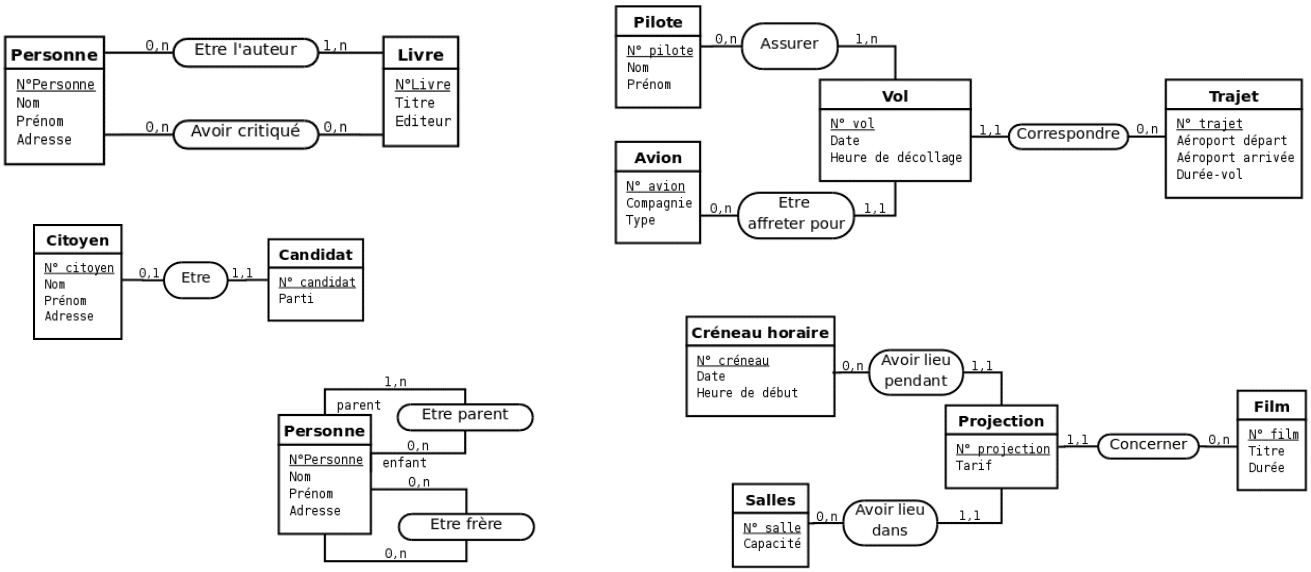
Modèle entité-association de la base garage

On peut ensuite lister toutes les caractéristiques des entités dont on aura besoin dans la base. Par exemple un client doit avoir un nom, prénom, adresse, numéro de téléphone et email pour pouvoir le contacter et éditer sa facture. Un véhicule doit avoir un numéro d'immatriculation, une marque et un type, …
On affine ainsi le schéma entité-association pour obtenir un **M**odèle **C**onceptuel des **D**onnées. Ce schéma ressemble beaucoup au schéma précédent, mais on a ajouté les attributs de chaque entité ainsi que la **cardinalité** de chaque association.

Les principales cardinalités sont les suivantes :

	Description	Exemple
0,1 ou 0..1	Une occurrence participe au moins 0 fois et au plus 1 fois à l'association	Un patient a 0 ou 1 médecin traitant
1,1 ou 1	Une occurrence participe exactement 1 fois à l'association	Un modèle de téléviseur a une et une seule marque
0,n ou 0..n	Une occurrence peut ne pas participer ou participer plusieurs fois	Un produit peut ne pas encore avoir été commandé, comme il peut l'avoir été plusieurs fois
1,n ou 1..n	Une occurrence participe au moins 1 fois, voire plusieurs	Un client commande au moins 1 produit

Quelques exemples de modèles conceptuels des données pour comprendre les cardinalités :



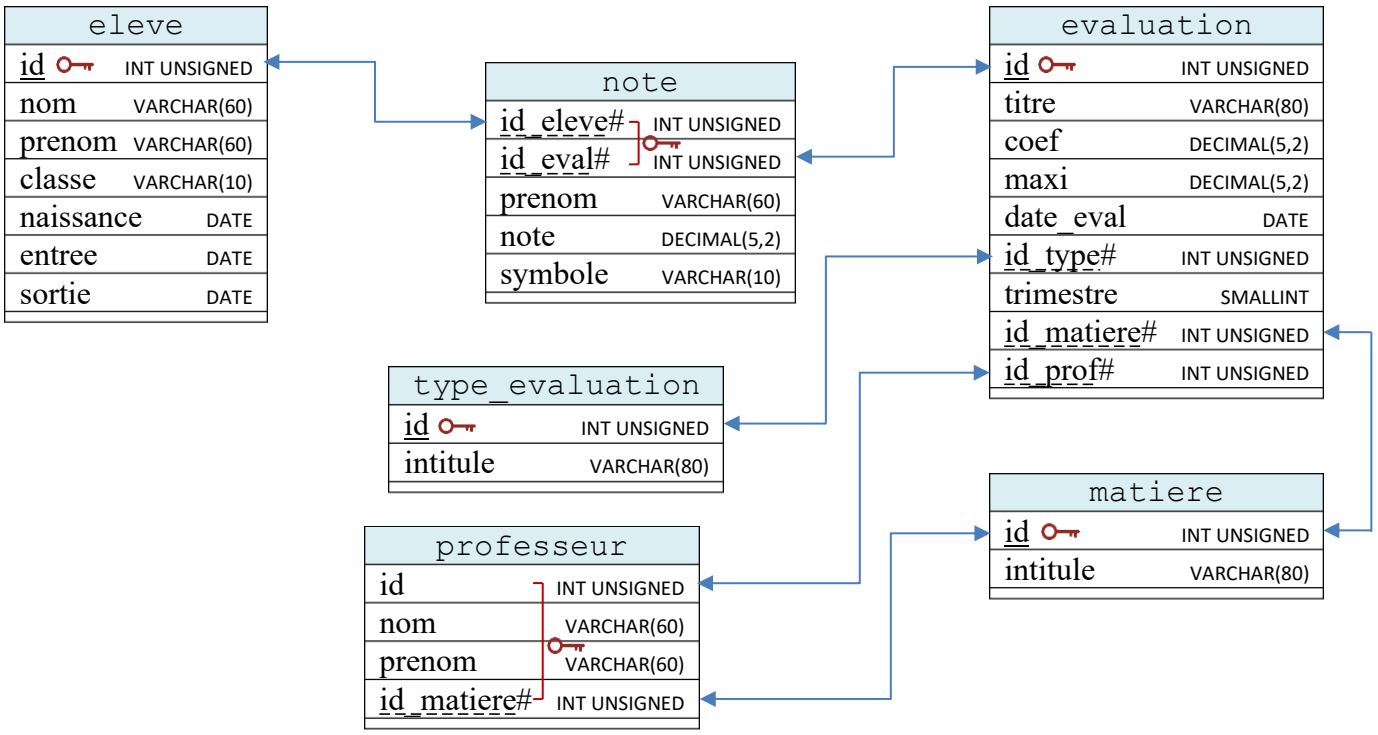
Application 4 :

Représenter un modèle conceptuel des données pour la base de données garage.

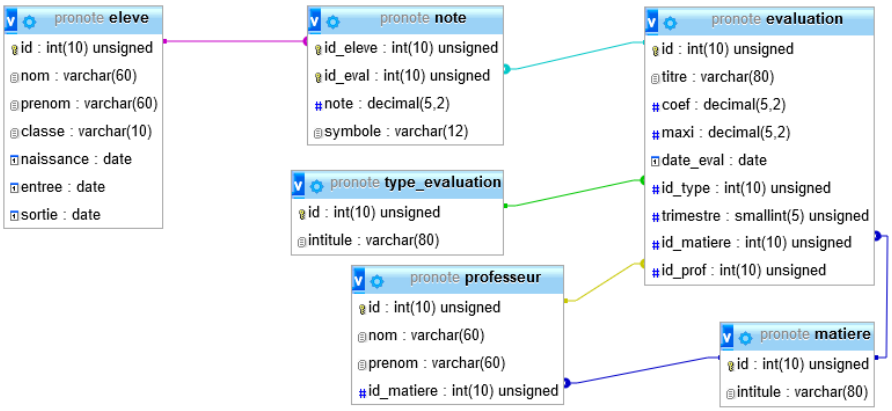
On remarque que dans ce modèle, chaque entité correspond à une relation du schéma relationnel. De même les caractéristiques des entités correspondent aux attributs des tables.

Pour se rapprocher encore du schéma relationnel de la base de données, on peut préciser le domaine de chaque attribut et transformer les associations en clés étrangères d'une table vers une autre. On obtient alors le **M**odèle **P**hysique des **D**onnées à partir duquel on peut écrire le schéma relationnel complet et définir complètement notre base de données.

Le modèle physique des données de notre base d'exemple pourrait donner ceci :



Ci-contre la vue affichée par phpmyadmin



4) Le langage SQL

Toute opération sur un SGBDR se fait par l'intermédiaire de commandes dans le langage SQL (pour ***S**tructured **Q**uery **L**anguage* ou langage requête structurée).
SQL est un **langage déclaratif**, c'est-à-dire que contrairement aux langages de programmation classique, il permet de *décrire* le résultat escompté, sans devoir indiquer la manière de l'obtenir. C'est ensuite le SGBD qui se charge de trouver la manière la plus adaptée pour calculer le résultat demandé (via un optimiseur de requête).
Le langage SQL a été normalisé en 1986 et a été mis à jour régulièrement. Globalement tous les SGBD respectent la norme du langage mais y ajoutent bien souvent des commandes propriétaires.
Le langage SQL n'est pas sensible à la casse, mais on prend souvent l'habitude d'écrire les instructions du langage en majuscule et les noms des tables et attributs en minuscule pour davantage de lisibilité.
De même on peut insérer autant d'espace que l'on veut entre les mots-clés et même des sauts de lignes. La fin d'une séquence d'instruction est matérialisée par un signe « `;` » comme en C ou en java.

5) Création d'une base de données

Pour créer et peupler une base de données (BD), on peut utiliser les commandes SQL suivantes :

Commande	Description
`CREATE DATABASE nom_de_la_base ;`	
`USE nom_de_la_base ;`	Permet de préciser la BD sur laquelle on va exécuter les commandes
`SHOW DATABASES ;`	Affiche toutes les BDs auxquelles on a accès

`SHOW TABLES ;`	Affiche la liste de toutes les relations de la BD
`CREATE TABLE schéma_relationnel ;`	Crée une nouvelle relation dont on doit spécifier le schéma relationnel entre parenthèses
`SHOW CREATE TABLE nom_table ;`	Affiche les informations sur la relation
`DROP TABLE nom_table ;`	Supprime la relation de la BD
`ALTER TABLE nom_table ;`	Modifie une relation existante
`INSERT INTO nom_table [(nom_1er_attribut, nom_2ème_attribut,…)] VALUES (valeur_attribut1, valeur_attribut2, …)[,(valeurs attributs n-uplet suivant)] ;`	Insère un ou plusieurs n-uplets dans une table.
`SELECT * FROM nom_table ;`	Visualiser les n-uplets d'une relation
`UPDATE table SET nom_attribut=valeur WHERE …;`	Modifie un ou plusieurs n-uplets
`DELETE FROM table WHERE nom_attribut=valeur;`	Supprime l'enregistrement de la table qui correspond à la condition du WHERE

Application 5 :

Pour cette application, on va utiliser UwAmp qui permet d'exécuter un serveur apache et mysql sous windows. Pour cela, localiser le répertoire de uwamp, puis dans une invite de commande, changer de répertoire pour aller sur `C:\Logiciels\UwAmp\bin\database\mysql-5.7.11\bin`.

De là on va exécuter l'interpréteur de commande de mysql en précisant le nom d'utilisateur (root) :

`mysql -u root -p` (le mot de passe est « root »)

Le prompt devient alors « `mysql >` » et on peut exécuter des commandes SQL. Pour avoir de l'aide à tout moment, on peut taper « `?` » afin d'avoir une liste des commandes, ou « `? nom_commande` » pour avoir de l'aide sur une commande particulière. Pour travailler en utf-8, taper les commandes « `CHARSET utf8 ;` » ou « `CHARSET utf8mb4 ;` ».

1) Créer une nouvelle base de données « `pronote` ». Créer les tables `eleve`, `evaluation` et `note`.

2) Peupler les tables avec quelques données et affichez-les.

Le fichier « pronote.sql » contient les commandes sql pour créer et peupler la base pronote.

3) Supprimer votre base (« `DROP DATABASE pronote ;` »), puis exécutez les commandes du fichier pronote.sql en utilisant la commande `SOURCE` de mysql.

4) Observer le fichier « pronote.sql ». Comment les contraintes de référence sont-elles déclarées ? D'après-vous à quoi servent `ON DELETE …` et `ON UPDATE …` ? Vérifier votre intuition en modifiant ou en supprimant des enregistrements.

A RETENIR :

- En apportant une vision plus abstraite de la table avec la notion de relation, le modèle relationnel permet de s'affranchir de la façon dont les données sont organisées et stockées en mémoire.
- Une base de données se compose de plusieurs relations afin de supprimer les redondances, faciliter les mises à jour et permettre les ajouts partiels.
- Les SGBD sont des logiciels complexes qui assurent toutes les fonctions liées à la création et l'utilisation d'une base de données. Il en existe de nombreux comme Oracle, mysql, postgresql ou microsoft SQL server.
- La structure d'une relation est donnée par son schéma relationnel qui précise le nom des attributs et leur domaine de valeurs.
- L'ensemble des enregistrements (ou n-uplets ou tuples) forme le corps de la relation.
- Dans chaque relation, l'un des attributs (parfois un groupe) dont la valeur permet d'identifier de manière unique un enregistrement, est appelé clé primaire.
- Certaines relations possèdent un attribut qui se réfère à la clé primaire d'une autre relation à laquelle elle est ainsi liée ; un tel attribut est appelé clé étrangère.
- Les contraintes d'intégrité (de domaine, de relation, de référence et utilisateur) assurent la cohérence des données. Le SGBD s'assure que toutes les opérations les vérifient.
- Le langage SQL (Structured Query Language) est un langage déclaratif plus ou moins standardisé avec lequel on peut donner des instructions au SGBD (créer, alimenter ou interroger la base de données).

Références :

Historique des SGBD : https://fadace.developpez.com/sgbdcmp/story/
Référence du langage SQL : https://sql.sh/ et https://www.w3bai.com/fr/sql/default.html
Cours : https://mermet.users.greyc.fr/Enseignement/EnseignementInformatiqueLycee/Havre/Bloc4/indexBD.html
https://laurent-audibert.developpez.com/Cours-BD/
http://www.bts-sio.com/cours/SI3/sql/0%20-%20Introduction%20aux%20SGBD%20(2016).php
Vidéos : https://www.lumni.fr/video/bases-de-donnees ,
https://www.youtube.com/watch?v=iu8z5QtDQhY (histoire des bases de données)
Logiciel de modélisation des données (tracé de schémas) : https://www.looping-mcd.fr/

Exercices – Bases de données relationnelles

Exercice 1 : Questions de cours

1) Peut ont choisir le nom d'un individu comme clé primaire (pour une relation 'élèves', par exemple) ?
2) Qu'est-ce qui permet de relier deux relations ?
3) Une clé primaire peut-elle comporter plusieurs attributs ?
4) Par qui le modèle relationnel a-t-il été créé ?
5) Une relation peut être vue comme :
   a. Une liste
   b. Un tableau à deux dimensions
   c. Une colonne
6) Parmi les affirmations suivantes, l'une n'est PAS nécessaire pour être une relation :
   a. Toutes les valeurs d'un attribut donné doivent être du même type
   b. Il ne doit pas y avoir deux tuples (lignes)) identiques dans la même relation
   c. Les attributs doivent être ordonnés
7) Un attribut est
   a. Une table en deux dimensions
   b. Une clé d'une relation
   c. Un tuple d'une relation
   d. Une colonne de la relation
8) Dans une relation, l'ordre des tuples (lignes) est-il important ?
9) Qu'est-ce qu'une clé primaire ?
10) Une clé étrangère c'est :
    a. Un attribut (ou un groupe d'attribut) d'une relation qui porte le même nom qu'un autre attribut dans une autre relation
    b. Un attribut (ou un groupe d'attribut) spécial présent dans chaque relation
    c. Un attribut (ou un groupe d'attribut) d'une relation qui est la clé primaire d'une autre relation
    d. Un attribut (ou un groupe d'attribut) d'une relation qui appartient en fait à une autre base de données

Considérons la base de données relationnelle schématisée ci-dessous :

SALARIES (Numsal, Nomsal, Ruesal, Cpsal, Villesal, Codeag)
- Clé primaire : Numsal
- Clé étrangère : Codeag en référence à Codeag de AGENCE

STAGE (Refstage, Nomstage, Duree)
- Clé primaire : Refstage

EFFECTUER (Numsal, Refstage)
- Clé primaire : Numsal + Refstage
- Clé étrangère 1 : Numsal en référence à Numsal de SALARIES
- Clé étrangère 2 : Refstage en référence à Refstage de STAGE

11) Quelle est la relation qui comprend deux attributs comme clé primaire ?
12) Toujours sur le même schéma de base de données, quelle est la clé primaire de la relation AGENCE ?
13) A partir de ce modèle relationnel un salarié peut-il faire plusieurs stages ?

CLIENT (Num_client, Nom_client, Rue, Ville, Code_postal, Telephone)
- Clé primaire : Num_client

FACTURE (Num-facture, Date_facture, Num_client)
- Clé primaire : Num-facture
- Clé étrangère : Num_client en référence à Num_client de CLIENT

14) A partir de ce modèle relationnel peut-on dire qu'une facture ne concerne qu'un et un seul client ?

Exercice 2 : Repérer les clés

Voici le schéma relationnel d'une base de données :

```
voyageur(id_voyageur, nom, prenom, ville, region)
sejour(id_sejour, id_voyageur#, code_logement#, debut, fin)
logement(code, nom, capacite, type, lieu)
activite(code_logement#, code_activite, description)
```

1) Combien y-a-t-il de relations ?
2) Citer toutes les clés primaires
3) Citer toutes les clés étrangères et expliquer leur utilité
4) Parmi les exemples ci-dessous, le(s)quel(s) pourrai(en)t être valide(s) dans le schéma relationnel donné ?

a.

```
voyageur = {(5,'Itemieu', 'Elmer', 'Clermont-Ferrand', 'Centre'), (8,'Terrieur', 'Alex',
'Amiens', 'Nord'), (13,'Stiké', 'Sophie', 'St-Malo', 'Bretagne')}
sejour = {(45,8,12,'2021-09-03','2021-09-10'), (2,8,12,'2021-09-07','2021-10-12'),
(45,2,7,'2020-05-07','2020-05-17')}
logement = {(1,'Les marmottes',8,'Chalet','St Veran'), (12,'Kili_5',6,'Appart','Arvieux')}
activite = {(12,5,'Ski'), (12,13,'Sauna'), (12,3,'Luge'), (1,5,'Ski'), (1,33,'Rando')}
```

b.

```
voyageur = {(9,'Nord', 'Léopold', 'Louvecienne', 'IdF'), (8,'Cover', 'Harry', 'Mulhouse',
'Grand-Est'), (9,'Honim', 'Anne', 'Bayonne', 'Occitanie') }
sejour = {(12,9,3,'2021-12-06','2021-12-10'), (9,8,5,'2021-15-07','2021-15-21'),
(5,9,5,'2020-07-09','2020-07-09')}
logement = {(3,'Les cabris-3',8,'Appart','Courchevel'), (5,'Alpen-8', 6, 'Appart', 'Lao')}
activite = {(5,1,'Snowboard'), (7,3,'Hammam'), (3,8,'Dodo'), (5,18,'Trek')}
```

c.

```
voyageur = {(14,'Diotie', 'Kelly', 'Briançon', 'Provence'), (22,'Camand', 'Mehdi', 'Troie',
'Bourgogne') ), (16,'Issier', 'Paul', 'Troie', 'Bourgogne')}
sejour = {(51,14,7,'2019-01-03','2019-01-09'), (27,22,4,'2021-12-18','2021-12-26')}
logement = {(7,'Skyjo',3,'Igloo','Maudane'), (4,'Chanterelles',4,'Appart','Maudane')}
activite = {(7,2,'Fond'), (4,3,'Piscine'), (7,8,'Kite'), (7,5,'Ciné'), (4,13,'Bar')}
```

Exercice 3 : Récupérer des données dans un modèle relationnel

Extrait de la table **`client`** :

IdClient	Nom	Adresse	Mail
1	Jean Bon	2 rue Jean Mermoz - Caen	jean.bon@free.fr
2	Alain Térieur	3 rue Paul Eluard - Hérouville	alain.terieur4@hotmail.com
3	Thérèse Etroit	16 rue de la porte - Mondeville	therese.etroit@orange.fr
4	Gilles Héjone	1 place de la bastille - Bénouville	gilledu14@sfr.fr
5	Hélène de Troie	2 rue Néper - Caen	ln23@laposte.net

Extrait de la table **`produit`** :

IdProduit	CodeProduit	NomProduit	Prix	Stock
1	12x24F	gel hydroalcoolique 100 ml	3,21	2
2	21s53R	masque FFP2 x100	10,57	3
3	97D74S	visière de protection	0,50	10
4	10F36A	désinfectant industriel 10L	53,25	5

Extrait de la table **`commande`** pour un jour *J* :

IdCmd	IdClient	IdProduit	Quantite	Expedie
1	4	1	1	1
2	2	2	2	0
3	1	1	3	1
4	3	3	4	0
5	1	4	3	0

1) Dans la table **`commande`** quelles est la clé primaire ? quelles sont les clés étrangères ?
2) A combien s'élève le montant total des commandes de la journée *J* ?
3) Quels sont les noms des clients pour lesquels la commande (ou une partie) a été expédiée ?
4) Quels sont les produits pour lesquels le stock ne sera pas suffisant ?

Exercice 4 : Définir le schéma d'une base

Une bibliothèque a une base de données composées de trois tables `auteurs`, `emprunteurs` et `emprunts`.
La relation Emprunteurs est définie par `emprunteurs(idEmp, Nom, Prenom, DateNaissance)`.

1) Pourquoi ni (`Nom`, `Prenom`), ni (`Nom`, `DateNaissance`) ne peuvent constituer des clés primaires ?
2) Quels sont les attributs possibles de la relation Auteurs ?
3) Quels sont les attributs possibles de la relation Emprunts ?

Exercice 5 : Structurer des données

On considère la table Films suivante :

Titre	Annee	Realisateur	Note	NbAavis	Duree	Type
Apocalypse Now	1979	Francis Ford Coppola	8,4	578150	2h27	Drama, Mystery, War
Full Metal Jacket	1987	Stanley Kubrick	8,3	644089	1h56	Drama, War
Scarface	1983	Brian De Palma	8,3	704025	170 min	Crime, Drama
Orange mécanique	1971	S. Kubrick	8,3	725846	136'	Crime, Drama, Sci-Fi
2001, l'odyssée de l'espace	1968	Kubrick	8,1	574171	2h29	Adventure, Science-fiction
Taxi Driver	1976	Martin Scorsese	8,3	685042	1h54	Crime, Drama

1) Proposer une structure conforme au modèle relationnel pour ces données.
2) Donner le contenu de chacune des tables.

Exercice 6 : Création de tables en SQL

Soit le modèle relationnel suivant :

```
Producteur(raison_sociale VARCHAR(25), ville VARCHAR(255))
Consommateur(login VARCHAR(10), email VARCHAR(50), nom VARCHAR(50), prenom VARCHAR(50),
ville VARCHAR(255))
Produit(id INTEGER, description VARCHAR(100), produit-par# VARCHAR(25)
(Producteur/Raison_sociale), consomme-par-login VARCHAR(10)(Consommateur/login),
consomme-par-email VARCHAR(50)(Consommateur/email))
Producteur(#raison_sociale VARCHAR(25), ville VARCHAR(255))
```

On ajoute que :

- `(nom,prenom,ville)` est une clé candidate de `Consommateur`
- Tous les produits sont produits
- Tous les produits ne sont pas consommés

1) Donner le modèle conceptuel des données de ce modèle relationnel.
2) Donner les commandes SQL permettant de créer les tables Producteur, Consommateur et Produit.
3) Donner les commandes SQL pour insérer les données correspondant aux assertions suivantes dans la base de données :
   - L'entreprise de Compiègne "Pommes Picardes SARL" a produit 4 lots de pommes, et 2 lots de cidre.
   - Il existe trois utilisateurs consommateurs dans la base, donc les adresses mails sont :
     Al.Un@compiegne.fr - Bob.Deux@compiegne.fr - Charlie.Trois@compiegne.fr
     Ce sont des employés de la ville de Compiègne qui habitent cette ville. Leur mail est construit sur le modèle Prenom.Nom@compiegne.fr. Leur login est leur prénom.
4) Donner les commandes SQL pour modifier les données de la base afin d'intégrer les assertions suivantes :
   - 1 lots de pommes a été consommés par Al Un.
   - 2 lots de pomme ont été consommé par Bob Deux.
   - Tous les lots de cidre ont été consommés par Al Un.
5) Charlie Trois n'ayant rien consommé, donner la commande SQL permettant de le supprimer de la base.

TNSI	Bases de données Bases de données relationnelles	Exercices – Bases de données	CORRIGE

Exercice 1 : Questions de cours

1) Nom, car il peut y avoir des homonymes (noms identiques).
2) Une clé étrangère.
3) Oui (exemple : Nom, Prenom, Date naissance)
4) Par E.F. Codd.
5) b.
6) c.
7) d.
8) Il n'a pas d'importance
9) C'est un groupe minimum d'attributs permettant d'identifier un tuple dans une relation.
10) c.
11) C'est la relation EFFECTUER qui a comme clé primaire Numsal et Refstage.
12) C'est Codeag qui est une clé étrangère de la relation SALARIES.
13) Un salarié peut faire plusieurs stages, mais pas plus d'une fois le même stage car Numsal et Refstage sont une clé primaire de EFFECTUER donc le même salarié (même valeur de Numsal) peut avoir plusieurs valeur de Refstage mais pas deux fois le même Refstage.
14) Chaque facture ne peut concerner qu'un seul client car il n'y a qu'un seul attribut Num_client dans la relation FACTURE et que Num-facture est une clé primaire, donc on ne peut avoir 2 n-uplets ayant le même numéro de facture mais des numéros de clients différents. En revanche, rien n'interdit qu'une facture n'ait pas de client (Num_client pris à NULL). Une facture peut donc concerner 0 ou 1 client.

Exercice 2 : Repérer les clés

Voici le schéma relationnel d'une base de données :

`voyageur(`**`id_voyageur`**`, nom, prenom, ville, region)`
`sejour(`**`id_sejour`**`, id_voyageur#, code_logement#, debut, fin)`
`logement(`**`code`**`, nom, capacite, type, lieu)`
`activite(`**`code_logement#`**`, `**`code_activite`**`, description)`

1) Il y a 4 relations (tables) : `voyageur`, `sejour`, `logement` et `activite`.
2) Les clés primaires sont en gras : **`id_voyageur`**, **`id_sejour`**, **`code`**, (**`code_logement`**, **`code_activite`**).
3) Les clés étrangères sont suivies d'un « # » : `id_voyageur` et `code_logement`. Elles servent à référencer un enregistrement dans une autre table (par exemple dans la table `sejour`, `code_logement` indique l'enregistrement de la table `logement` qui correspond à ce séjour.
4) a. ne peut pas être valide, car la table `sejour` contient 2 séjours ayant la même référence (45), ce qui viole la contrainte de relation (unicité de la clé primaire).
   b. ne peut pas être valide car il y a deux voyageurs ayant la même clé primaire (9). De plus l'activité « Hammam » a pour code logement 7 et il n'y a pas de logement ayant ce numéro dans la relation `logement` ce qui viole la contrainte de référence.
   c. Cette séquence est valide et respecte le schéma relationnel.

Exercice 3 : Récupérer des données dans un modèle relationnel

1) Dans la table **`commande`** la clé primaire est IdCmd, les clés étrangères sont IdClient (table client), IdProduit (table produit) ?
2) Pour la journée *J*, on a 1 gel pour 3,21 € + 2 masques pour 2×10,57 = 21,14 € + 3 gels pour 3 × 3,21 = 9,63 € + 4 visières pour 4 × 0,5 = 2 € + 3 désinfectants pour 3 × 53,25 = 159,75 € soit un total de **195,73 €**.
3) Les clients 4 (Gilles Héjone) et 3 (Thérèse Etroit) ont bénéficié d'une expédition.
4) Le stock sera insuffisant pour le gel car la commande 1 en consomme 1 et il en reste 2-1 = 1 en stock et la commande 3 en demande 3 alors qu'il n'en reste qu'un seul (c'est pour ça qu'on en expédie qu'un). Pour les autres articles, le stock est suffisant.

Exercice 5 : Structurer des données

1) Proposer une structure conforme au modèle relationnel pour ces données.

On constate que le même réalisateur apparaît plusieurs fois avec des écritures légèrement différentes et même chose pour les types. On va donc créer des tables séparées pour éviter cette redondance et éviter les différences d'écriture dans le nom du réalisateur ou le libellé du type.

On pourrait créer 4 tables :
film (id_film INT UNSIGNED, titre VARCHAR(60), annee INT, id_realisateur# INT UNSIGNED, note DECIMAL(2,1), nb_avis INT UNSIGNED, duree TIME)
realisateur (id_realisateur INT UNSIGNED, nom VARCHAR(30), prenom VARCHAR(30))
type (id_type INT UNSIGNED, libele VARCHAR(30))
type_film (id_film# INT UNSIGNED, id_type# INT UNSIGNED)

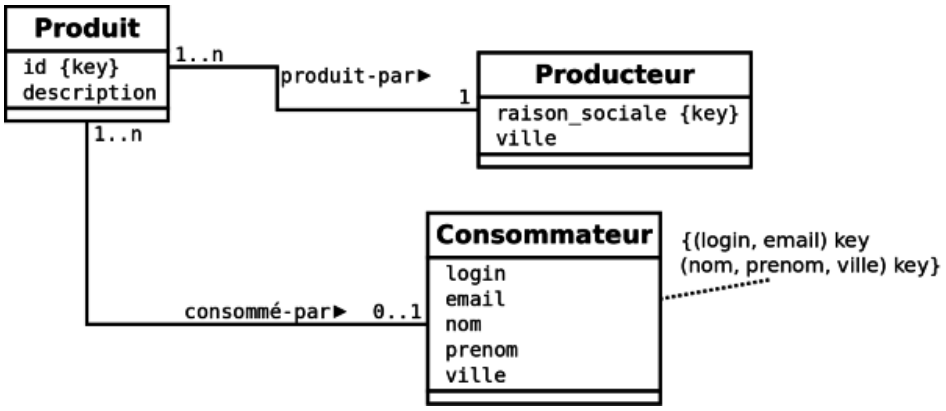
2) Donner le contenu de chacune des tables.

On aurait :

```
film = {(1,'Apocalipse Now',1979,1,8.4,578150,'02:27'),
     (2,'Full Metal Jacket',1987,2,8.3,644089,'01:56'), (3,'Scarface',1983,3,8.3,704025,'02:50'),
     (4,'Orange mécanique',1971,2,8.3,725846,'02:16'), (6,'Taxi Driver',1976,4,8.3,685042,'01:54'),
     (5,'2001, l'odyssée de l'espace',1968,3,8.1,574171,'02:29')}
realisateur = {(1,'Coppola','Francis Ford'), (2,'Kubrick','Stanley'), (3,'De Palma','Brian'),
               (4,'Scorsese','Martin')}
type = {(1,'Drama'),(2,'Mystery'),(3,'War'),(4,'Crime'),(5,'Sci-Fi'),(6,'Adventure')}
type_film={(1,1),(1,2),(1,3), (2,1),(2,3), (3,4), (3,1), (4,4),(4,1),(4,5), (5,6),(5,5), (6,4),(6,1)}
```

Exercice 6 : Création de tables en SQL

1)



2)

```
CREATE TABLE  Producteur (
raison_sociale VARCHAR (25),
ville VARCHAR(255),
PRIMARY KEY (raison_sociale));

CREATE TABLE Consommateur (
```

```
login VARCHAR(10),
email VARCHAR(50),
nom VARCHAR(50) NOT NULL,
prenom VARCHAR(50) NOT NULL,
ville VARCHAR(255) NOT NULL,
PRIMARY KEY (login,email),
UNIQUE (nom,prenom,ville));

CREATE TABLE Produit (
id INTEGER,
description VARCHAR(100),
produit_par VARCHAR(25) NOT NULL,
consomme_par_login VARCHAR(10),
consomme_par_email VARCHAR(50),
PRIMARY KEY (id),
FOREIGN KEY (produit_par) REFERENCES Producteur(raison_sociale),
FOREIGN KEY (consomme_par_login,consomme_par_email) REFERENCES Consommateur(login,email));
```

3)

```
-- Insertion du producteur
INSERT INTO Producteur (raison_sociale, ville)
VALUES ('Pommes Picardes SARL', 'Compiègne');

-- Insertion des produits
INSERT INTO Produit (id, description, produit_par)
VALUES (1, 'Lot de pommes', 'Pommes Picardes SARL');

INSERT INTO Produit (id, description, produit_par)
VALUES (2, 'Lot de pommes', 'Pommes Picardes SARL');

INSERT INTO Produit (id, description, produit_par)
VALUES (3, 'Lot de pommes', 'Pommes Picardes SARL');

INSERT INTO Produit (id, description, produit_par)
VALUES (4, 'Lot de pommes', 'Pommes Picardes SARL');

INSERT INTO Produit (id, description, produit_par)
VALUES (5, 'Lot de cidre', 'Pommes Picardes SARL');

INSERT INTO Produit (id, description, produit_par)
VALUES (6, 'Lot de cidre', 'Pommes Picardes SARL');

-- Insertion des consommateurs
INSERT INTO Consommateur (login, email, nom, prenom, ville)
VALUES ('Al', 'Al.Un@compiegne.fr', 'Un', 'Al', 'Compiègne');

INSERT INTO Consommateur (login, email, nom, prenom, ville)
VALUES ('Bob', 'Bob.Deux@compiegne.fr', 'Deux', 'Bob', 'Compiègne');

INSERT INTO Consommateur (login, email, nom, prenom, ville)
VALUES ('Charlie', 'Charlie.Trois@compiegne.fr', 'Trois', 'Charlie', 'Compiègne');
```

4)

```
UPDATE produit
SET consomme_par_login='Al', consomme_par_email='Al.Un@compiegne.fr'
WHERE id=1;

UPDATE produit
SET consomme_par_login='Bob', consomme_par_email='Bob.Deux@compiegne.fr'
WHERE id=2 OR id=3;

UPDATE produit
SET consomme_par_login='Al', consomme_par_email='Al.Un@compiegne.fr'
WHERE description='Lot de cidre';
```

5)

```
DELETE FROM consommateur WHERE login='Charlie' AND email='Charlie.Trois@compiegne.fr';
```