Introduction

L'informatique est la science du traitement automatique de l'information. Son développement dans tous les secteurs de nos sociétés s'accompagne de la mise en place de gigantesques bases données. Celles-ci sont gérées par des Systèmes de Gestion de Bases de Données (SGBD) qui doivent répondre à de nombreuses problématiques : persistance, sécurité, personnalisation des droits d'accès, facilité d'interrogation pour les humains ou les programmes (API), accès concurrents, garantie de l'intégrité des données.
Après une présentation des principales fonctionnalités des SGBD, nous présenterons le modèle relationnel qui est le principal modèle logique de structuration des données utilisé par les SGBD depuis les années 1980.

1 Base de données et SGBD

1.1 Base de données

Définition 1

Une **base de données** est un ensemble d'informations structurées qui sont mémorisées sur un support persistant.

Exemple 1

Un fichier CSV est un exemple élémentaire de base de données. Ci-dessous on a représenté l'extrait d'un fichier CSV où chaque ligne représente un ordinateur portable avec une séquence de valeurs séparées par des virgules, correspondant aux attributs listés sur la première ligne.

- chaque **ligne** ou **enregistrement** décrit une même entité (un ordinateur portable dans l'exemple du graphique 1) ;
- chaque **colonne** représente un **descripteur** ou **attribut** qui prend ses valeurs dans un certain **domaine** : par exemple le portable de numéro de série `N1J4HSMY` a été acheté en 2021 et son système d'exploitation est `ubuntu` en version `18.04`.

```
num_serie,nom_poste,nom_modele,fabricant,annee,nom_systeme,version
S2069FST,Portable-01,hp probook 400,hp,2019,fedora linux,36
JQ192AFH,PORTABLE-02,latitude e7470,dell,2018,debian,bullseye
N1J4HSMY,PORTABLE-03,hp probook 600,hp,2021,ubuntu,18.04
GENS27BP,PORTABLE-04,hp probook 400,hp,2020,debian,buster
PW5A4Z5Z,Portable-05,lenovo x270,lenovo,2017,debian,buster
ZM067CMA,portable-06,hp probook 600,hp,2021,debian,bullseye
```

Nous avons vu en première comment manipuler en Python ces tables de données par extraction dans une structure comme un tableau de dictionnaires :

```
table = [
    {
        "num_serie": "S2069FST",
        "nom_poste": "Portable-01",
        "nom_modele": "hp probook 400",
        "fabricant": "hp",
        "annee": "2019",
        "nom_systeme": "fedora linux",
        "version": "36",
    },
    {
        "num_serie": "JQ192AFH",
        "nom_poste": "PORTABLE-02",
        "nom_modele": "latitude e7470",
        "fabricant": "dell",
        "annee": "2018",
        "nom_systeme": "debian",
        "version": "bullseye",
    },
]
```

 **Limitations :**

L'utilisation de tels fichiers plats où les données sont stockées de façon séquentielle, n'est pas adaptée pour la gestion de bases de données de grande taille :

- *l'accès direct au fichier stockant les données pose des problèmes d'intégrité (modification par erreur), de sécurité (modification malveillante) et de gestion de l'accès concurrent (plusieurs utilisateurs en parallèle) ;*
- *les performances en lecture/écriture sont insuffisantes.*

1.2 Système de Gestion de Bases de Données

Définition 2

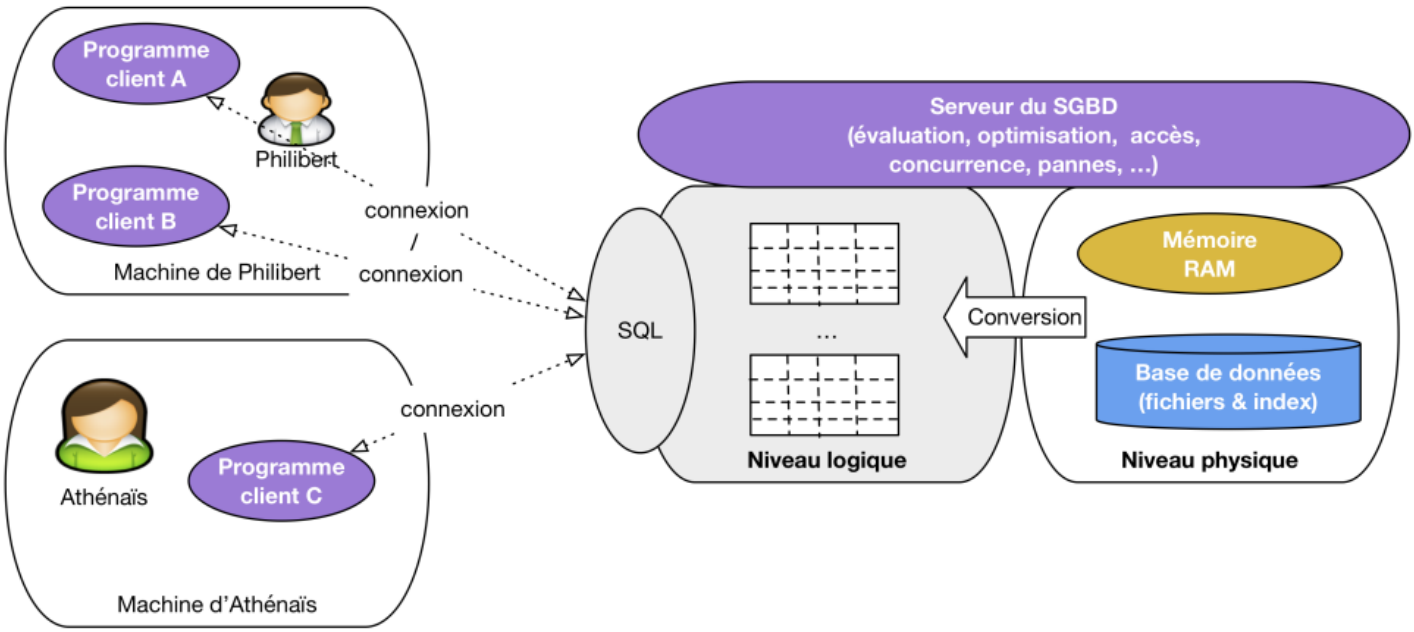
Un **Système de Gestion de Bases de Données** (SGBD) est un système informatique qui assure la gestion de l'ensemble des informations stockées dans une base de données. Il prend en charge :

- ☞ le *contrôle d'accès* aux fichiers de la base en gérant les accès concurrents ;
- ☞ la garantie de la *persistance* et de l'*intégrité* des données ;
- ☞ les *interactions* avec les applications et les utilisateurs, grâce à des langages d'interrogation et de manipulation de données comme SQL, en optimisant les requêtes et le stockage des données.

Un SGBD contrôle l'accès aux fichiers physiques de la base de données à travers une architecture *client / serveur*.
Dans un SGBD, une opération sur les données s'appelle une **transaction**. Elle doit vérifier quatre propriétés ACID :

- *Atomicité* : une transaction se fait au complet ou pas du tout ;
- *Cohérence* : une transaction doit laisser la base de données dans un état cohérent ;
- *Isolation* : si deux transactions s'exécutent simultanément, il ne doit pas exister de dépendance entre les deux, l'état de la base de données doit être le même que si elles s'étaient exécutées séquentiellement ;
- *Durabilité* : si une transaction a été confirmée, elle demeure enregistrée même en cas de panne électrique.

Un SGBD offre une interface publique de représentation de la base de données sous la forme d'un **modèle logique**. Il doit exister une indépendance entre les niveaux logique et physique (concept d'**encapsulation**).



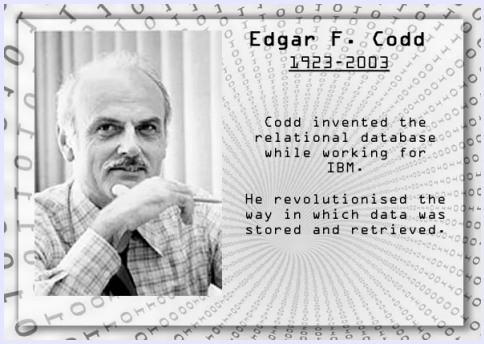
Source : cours de Philippe Rigaux.

2 Le modèle relationnel

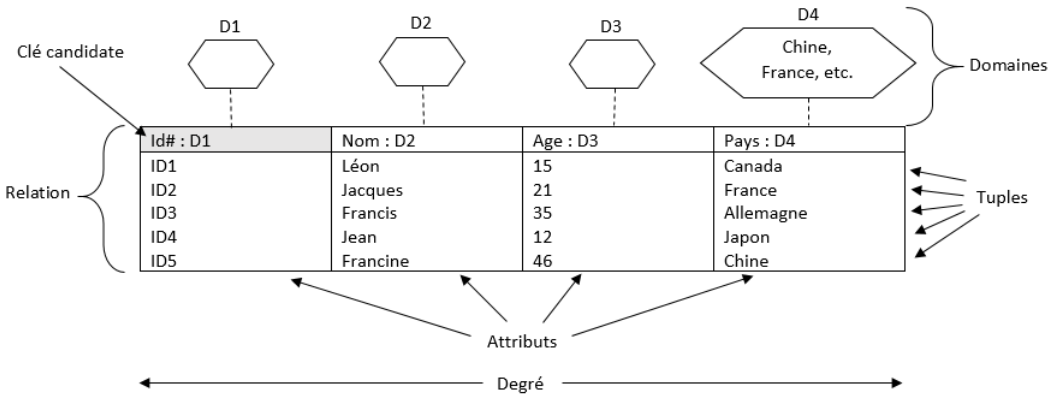
2.1 Histoire

Histoire 1

Le principal modèle logique de représentation des données utilisé dans les SGBD est le *modèle relationnel.*
Il est apparu en 1970 dans un article de E.F. Codd chercheur chez IBM. E.F. Codd décrit une représentation abstraite d'une base de données fondée sur deux concepts élémentaires : la *relation* et le *domaine.*



Une **relation** est un ensemble de **nuplets**, représentant des entités du monde réel (par exemple un ordinateur portable) prenant leurs valeurs dans des **domaines** (numéro de série, nom, fabricant, date de livraison …). On peut représenter une **relation** par une **table** et un **nuplet** par une **ligne**. La force du **modèle relationnel** réside dans sa simplicité et sa rigueur mathématique : des opérateurs algébriques et logiques permettent de formuler les transactions sur la base.
Ce modèle issu de la recherche scientifique s'impose progressivement auprès des industriels au cours des années 1970 et en 1979 Oracle publie un premier SGBD relationnel commercial.
Dès 1970, IBM, employeur de E.F. Codd, lance Sytem R un projet de recherche sur un SGBD relationnel. Ce projet montre l'efficacité du modèle relationnel et le langage d'interrogation et de manipulation de données SEQUEL qui est développé, deviendra ensuite SQL. Ce langage s'imposera, pour les SGBD relationnels, comme une norme dès 1986 .
Ces derniers dominent le marché depuis 1980. A partir des années 2010, leur domination est remise en cause par les géants de l'internet qui ont besoin de bases de données distribuées dans de multiples centres de données, avec une priorité mise sur l'accessibilité plutôt que sur la cohérence des données. Ils développent en interne des SGBD non relationnels dits NoSQL.



Structure d'une relation, source : Wikipedia

2.2 Table ou relation

Définition 3 *Table ou relation*

Dans le *modèle relationnel*, une base de données est organisée sous la forme d'une ou plusieurs **tables**.

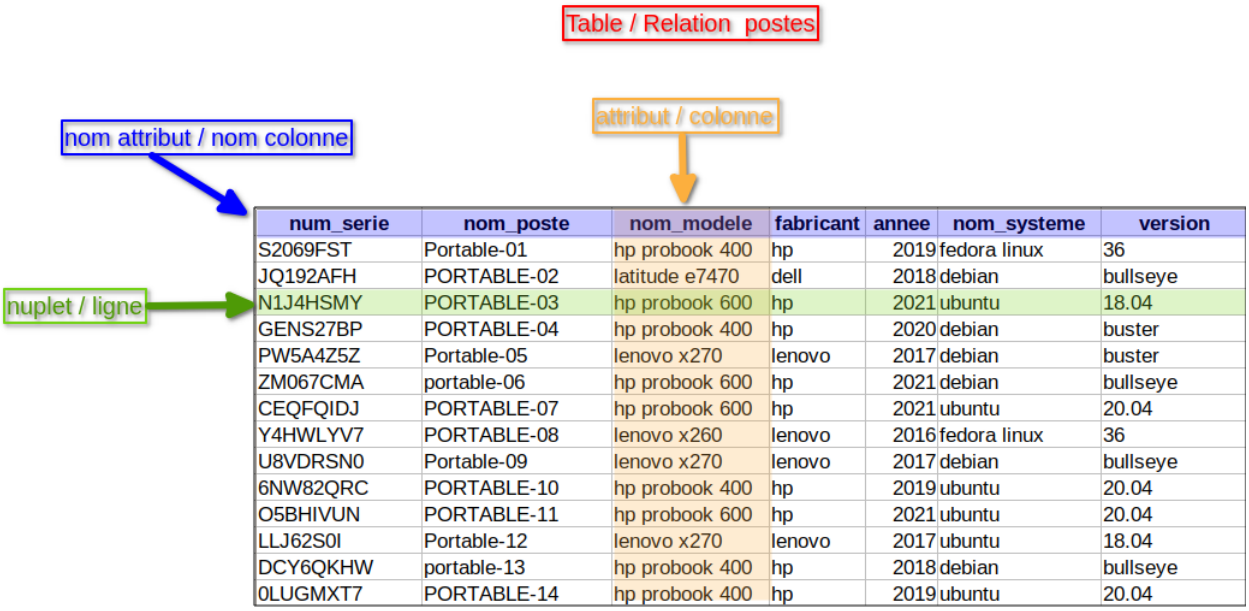
- ☞ Une **table** est un ensemble (non ordonné) de **lignes**. Une **table** contient des données sur des *entités* du monde réel de même nature.
- ☞ Chaque **ligne** représente un *entité* sous la forme d'une séquence (ordonnée) de valeurs, chacune d'un **type** bien déterminé.
- ☞ L'ensemble des valeurs de même type constitue une **colonne** caractérisée par un nom de colonne.

On peut donner une formalisation mathématique du *modèle relationnel* : une base de données est organisée sous la forme d'une ou plusieurs **relations**.

- ☞ Une **relation** est un ensemble (non ordonné) de **nuplets**. Une **relation** contient des données sur des *entités* du monde réel qui sont de même nature.
- ☞ Chaque **nuplet** représente une *entité* sous la forme d'une séquence (ordonnée) de valeurs, chacune prend ses valeurs dans un **domaine** bien déterminé.
- ☞ L'ensemble des valeurs d'un même domaine constitue un **attribut** de la relation.

Représentation par relation	Représentation par table
Relation	Table
Nuplet	Ligne
Nom d'attribut	Nom de colonne
Valeur d'attribut	Cellule
Domaine	Type

Voici un exemple de **table / relation** :



num_serie	nom_poste	nom_modele	fabricant	annee	nom_systeme	version
S2069FST	Portable-01	hp probook 400	hp	2019	fedora linux	36
JQ192AFH	PORTABLE-02	latitude e7470	dell	2018	debian	bullseye
N1J4HSMY	PORTABLE-03	hp probook 600	hp	2021	ubuntu	18.04
GENS27BP	PORTABLE-04	hp probook 400	hp	2020	debian	buster
PW5A4Z5Z	Portable-05	lenovo x270	lenovo	2017	debian	buster
ZM067CMA	portable-06	hp probook 600	hp	2021	debian	bullseye
CEQFQIDJ	PORTABLE-07	hp probook 600	hp	2021	ubuntu	20.04
Y4HWLYV7	PORTABLE-08	lenovo x260	lenovo	2016	fedora linux	36
U8VDRSN0	Portable-09	lenovo x270	lenovo	2017	debian	bullseye
6NW82QRC	PORTABLE-10	hp probook 400	hp	2019	ubuntu	20.04
O5BHIVUN	PORTABLE-11	hp probook 600	hp	2021	ubuntu	20.04
LLJ62S0I	Portable-12	lenovo x270	lenovo	2017	ubuntu	18.04
DCY6QKHW	portable-13	hp probook 400	hp	2018	debian	bullseye
0LUGMXT7	PORTABLE-14	hp probook 400	hp	2019	ubuntu	20.04

Graphique 1

Définition 4 *Schéma relationnel*

Une relation peut être décrite par :

- ☞ son **nom** ;
- ☞ un nom distinct pour chaque **attribut** ;
- ☞ le **domaine** de valeur de chaque attribut.

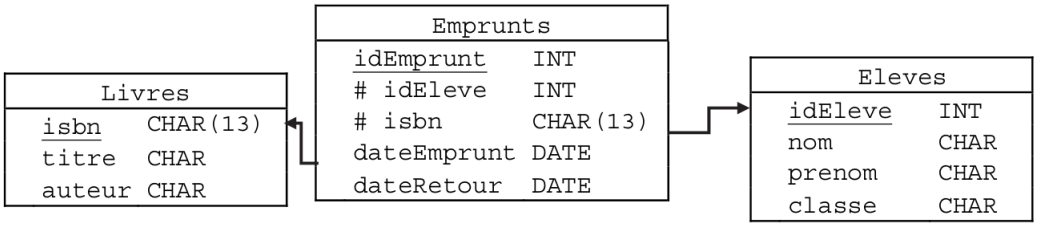
Cette description peut être résumée dans un **schéma relationnel**.
Par exemple la **table** du graphique 1 est une **relation**, nommons la *postes*, de schéma :

```
postes(num_serie: TEXT, nom_poste: TEXT, nom_modele: TEXT, fabricant: TEXT,
      annee: INT, nom_systeme: TEXT, version: TEXT)
```

Le **schéma relationnel d'une base** constituée de plusieurs relations est l'ensemble des schémas de ces relations. On peut le représenter sous la forme d'un diagramme faisant apparaître les dépendances entre les différentes relations.
Par exemple, la base représentée ci-dessous a pour schéma :

- Livres(isbn : CHAR(13), titre : CHAR, auteur : CHAR)
- Emprunts(idEmprunt : INT, # idEleve : INT, # isbn : CHAR(13), dateEmprunt : DATE, dateRetour : DATE)
- Eleves(idEleve : INT, nom : CHAR, prenom : CHAR, classe : CHAR)

Les attributs soulignés ou préfixés avec un # sont des **clefs** (voir section 2.4).



Exercice 1 : *Asie juin 2022 Partie 1*

On souhaite gérer un club de tennis en ligne avec la possibilité de réserver un terrain à un créneau horaire. Le site ne gère que des réservations pour des matchs en simple. Voici la structure de la base de données :

- Relation contenant l'ensemble des joueurs du club avec leurs identifiants.

joueurs				
id_joueur	**nom_joueur**	**prenom_joueur**	**login**	**mdp**
1	Dupont	Alice	alice	1234
2	Durand	Belina	belina	5694
3	Caron	Camilia	camilia	9478
4	Dupont	Dorine	dorine	1347

- Relation précisant les matchs joués.

matchs					
id_match	**date**	**id_creneau**	**id_terrain**	**id_joueur1**	**id_joueur2**
1	2020-08-01	2	1	1	4
2	2020-08-01	3	1	2	3
3	2020-08-02	6	2	1	3
4	2020-08-02	7	2	2	4
5	2020-08-08	3	3	1	2
6	2020-08-08	5	2	3	4

- Relation précisant les différents terrains.

terrains		
id_terrain	**nom_terrain**	**surface**
1	stade	terre battue
2	gymnase	synthétique
3	hangar	terre battue

- Relation précisant les créneaux réservables.

creneaux	
id_creneau	**plage_horaire**
1	8h-9h
2	9h-10h
3	10h-11h
4	11h-12h
5	12h-13h
6	13h-14h
7	14h-15h
8	15h-16h
9	16h-17h
10	17h-18h
11	18h-19h
12	19h-20h

On donne les types de données disponibles sur le SGBD :

Domaine	Caractéristique
`CHAR(t)`	Texte fixe de t caractères de taille fixe (bourrage avec des espaces si moins de t caractères)
`VARCHAR(t)`	Texte d'au plus t caractères de taille variable (pas de bourrage avec des espaces)
`TEXT`	Texte de 65 535 caractères max.
`INT`	Nombre entier de -2^{31} à $2^{31}-1$ (signé) ou de 0 à $2^{32}-1$ (non signé)
`FLOAT`	Réel à virgule flottante
`DATE`	Date format AAAA-MM-JJ
`DATETIME`	Date et heure format AAAA-MM-JJHH :MI :SS

Détermine le schéma relationnel de cette base de données.

.....

.....

.....

.....

.....

.....

2.3 Qualité d'une base de données relationnelle

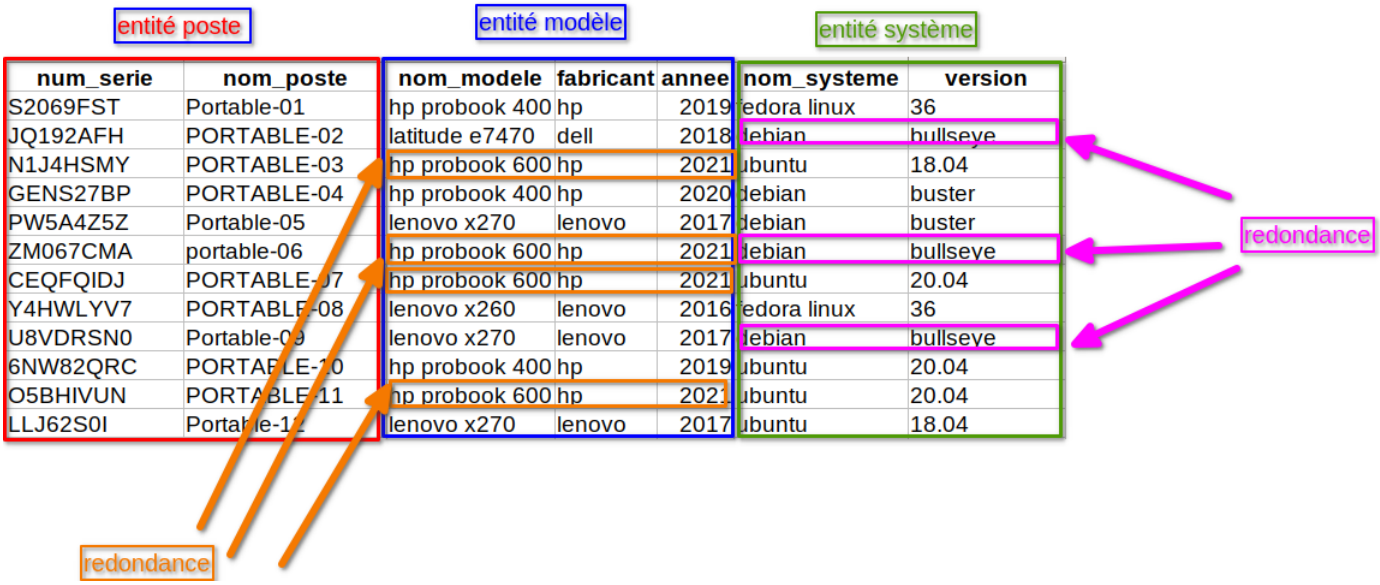
Propriété 1 *Qualité d'un schéma relationnel*

Un **schéma relationnel** d'une base de données doit vérifier certains principes :

- ☞ *Unicité / contrainte de relation* : chaque **relation** ne peut contenir deux **nuplets** identiques ;
- ☞ *Contrainte de domaine* : la valeur d'un **attribut** doit appartenir au **domaine** de l'attribut ;
- ☞ *Atomicité* : une valeur d'un **attribut** est une donnée atomique, elle ne peut contenir d'autres données. Les **types** des attributs sont donc simples (entier, flottant, chaînes de caractères, booléens ...) et ne peuvent être structurés (listes ...) ;
- ☞ *Non redondance* : une donnée ne doit être représentée qu'une fois dans une table et des données identiques ne doivent pas se trouver dans des tables différentes. Une entité du monde réel ne doit être représentée que par un seul **nuplet**.

Exercice 2 : *Normalisation d'un schéma avec redondances*

On considère de nouveau la relation présentée sous forme de table dans le graphique 1 page 6.



num_serie	nom_poste	nom_modele	fabricant	annee	nom_systeme	version
S2069FST	Portable-01	hp probook 400	hp	2019	fedora linux	36
JQ192AFH	PORTABLE-02	latitude e7470	dell	2018	debian	bullseye
N1J4HSMY	PORTABLE-03	hp probook 600	hp	2021	ubuntu	18.04
GENS27BP	PORTABLE-04	hp probook 400	hp	2020	debian	buster
PW5A4Z5Z	Portable-05	lenovo x270	lenovo	2017	debian	buster
ZM067CMA	portable-06	hp probook 600	hp	2021	debian	bullseye
CEQFQIDJ	PORTABLE-07	hp probook 600	hp	2021	ubuntu	20.04
Y4HWLYV7	PORTABLE-08	lenovo x260	lenovo	2016	fedora linux	36
U8VDRSN0	Portable-09	lenovo x270	lenovo	2017	debian	bullseye
6NW82QRC	PORTABLE-10	hp probook 400	hp	2019	ubuntu	20.04
O5BHIVUN	PORTABLE-11	hp probook 600	hp	2021	ubuntu	20.04
LLJ62S0I	Portable-12	lenovo x270	lenovo	2017	ubuntu	18.04

Graphique 2

- Cette relation respecte bien le *principe d'unicité* car chaque **nuplet** est identifié de façon unique par la valeur de l'attribut `num_serie` ou même par la valeur de l'attribut `nom_poste`.

  *Un attribut qui identifie de façon unique un **nuplet** s'appelle une **clef**. S'il est minimal on parle de **clef primaire**.*

- En revanche la relation présente de nombreuses *redondances*. Ces répétitions sont liées au fait qu'un **nuplet** ne contient pas des données sur une seule mais sur trois entités distinctes du monde réel :

  – les **attributs** `num_serie` et `nom_poste` caractérisent une entité *poste informatique*;
  – les **attributs** `nom_modele`, `fabricant` et `annee` caractérisent une entité *modèle de poste*;
  – les **attributs** `nom_systeme` et `version` caractérisent une entité *système d'exploitation*.

- **Question 1 :** Donner quelques exemples d'anomalies que peut entraîner la redondance d'informations pour une relation. On considérera les situations suivantes : on insère un nouveau nuplet, on modifie une valeur d'attribut redondant, on supprime un nuplet.

  .....

  .....

  .....

  .....

- Pour éliminer les redondances, ont peut séparer la base de données en trois relations, une par entité, en éliminant les doublons dans les relations `modele` et `systeme`

Relation `poste`

num_serie	nom_poste
S2069FST	Portable-01
JQ192AFH	PORTABLE-02
N1J4HSMY	PORTABLE-03
GENS27BP	PORTABLE-04
PW5A4Z5Z	Portable-05
ZM067CMA	portable-06
CEQFQIDJ	PORTABLE-07
Y4HWLYV7	PORTABLE-08
U8VDRSN0	Portable-09
6NW82QRC	PORTABLE-10
O5BHIVUN	PORTABLE-11
LLJ62S0I	Portable-12

Relation `modele`

nom_modele	fabricant	annee
lenovo x260	lenovo	2016
lenovo x270	lenovo	2017
latitude e7470	dell	2018
hp probook 400	hp	2018
hp probook 400	hp	2019
hp probook 400	hp	2020
hp probook 600	hp	2021

Relation `systeme`

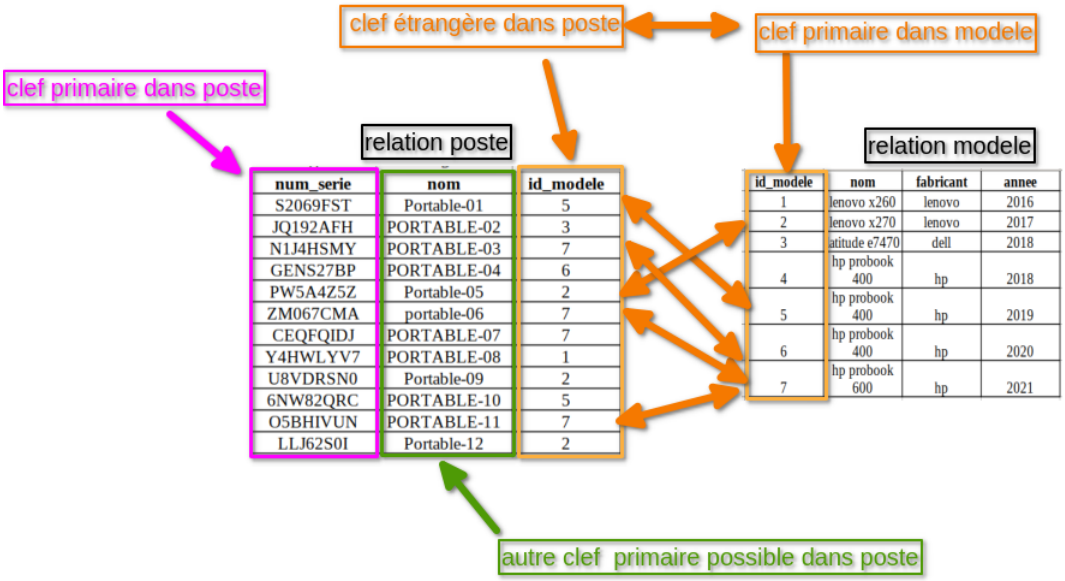
nom_systeme	version
debian	bullseye
debian	buster
fedora linux	35
fedora linux	36
ubuntu	18.04
ubuntu	20.04

Graphique 3

Cependant il est impossible dans ce cas de reconstituer l'information complète sur un poste avec son modèle et son système d'exploitation.

Pour relier une ligne de la table `poste` et une ligne de la la table `systeme` qui lui correspond, on va étendre chaque relation avec un attribut qui prend la même valeur. De plus pour associer à la ligne de `poste`, l'unique ligne de `modele` qui lui correspond, cet attribut doit être un identifiant unique dans la table `modele`, c'est-à-dire une **clef primaire**. Pour marquer la correspondance, on a choisi de donner le même nom `id_modele` à ces deux attributs mais ce n'est pas obligatoire. Dans la table `poste`, l'attribut `id_modele` n'est pas une clef primaire car un même modèle peut

être associé à plusieurs postes mais comme il référence une **clef primaire** d'une autre table, on le qualifie de **clef étrangère**.



num_serie	nom	id_modele
S2069FST	Portable-01	5
JQ192AFH	PORTABLE-02	3
N1J4HSMY	PORTABLE-03	7
GENS27BP	PORTABLE-04	6
PW5A4Z5Z	Portable-05	2
ZM067CMA	portable-06	7
CEQFQIDJ	PORTABLE-07	7
Y4HWLYV7	PORTABLE-08	1
U8VDRSN0	Portable-09	2
6NW82QRC	PORTABLE-10	5
O5BHIVUN	PORTABLE-11	7
LLJ62S0I	Portable-12	2

id_modele	nom	fabricant	annee
1	lenovo x260	lenovo	2016
2	lenovo x270	lenovo	2017
3	atitude e7470	dell	2018
4	hp probook 400	hp	2018
5	hp probook 400	hp	2019
6	hp probook 400	hp	2020
7	hp probook 600	hp	2021

Graphique 4

- **Question 2 :** Compléter les relations `poste` et `systeme` avec des attributs permettant de retrouver pour un poste le système d'exploitation qui est installé. Donner le schéma de la base de données constitué par les trois relations `poste`, `modele` et `systeme`. On soulignera les **clefs primaires** et préfixera du caractère # les **clefs étrangères**.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

*La définition de clefs primaires et étrangères permettant d'éliminer les doublons et les redondances constituent une étape de **normalisation**, permettant un bon schéma relationnel pour une base de données. Le modèle relationnel assure que cette normalisation est toujours possible. La détection d'anomalies est au programme mais pas la conception de bases de données.*

2.4 Clef primaire, clef étrangère

Définition 5 *Clef d'une relation*

☞ Étant donné une relation R, une **clef** de R est un attribut de R tel que :

- **Condition 1 :** l'attribut a une valeur pour tout nuplet de la relation R ;
- **Condition 2** il ne peut exister deux nuplets de R avec la même valeur pour cet attribut.

Si on parle de table, une **clef** d'une table est une colonne qui ne contient pas valeur non renseignée et telle qu'à un instant donné, il ne peut exister plus d'une ligne dans la table ayant une valeur fixée pour cette colonne.

☞ Une clef peut être constituée d'un ensemble de plusieurs attributs (un ensemble de plusieurs colonnes).

☞ La connaissance de la valeur des attributs d'une clef suffit à déterminer complètement le nuplet (la ligne) dans la relation (la table).

☞ Pour respecter la contrainte d'unicité, une relation doit posséder au moins une clef (l'ensemble des attributs par exemple), elle peut en posséder plusieurs.

☞ Le choix d'une clef minimale en nombre d'attributs détermine une **clef primaire** pour la relation.

☞ Dans le modèle relationnel, une **clef primaire** doit être déclarée dans le schéma lors de la création de la base de données car elle constitue une **contrainte d'unicité** garantissant l'absence de doublons dans une relation.

Une relation doit comporter une clef primaire et on ne peut définir qu'une seule clef primaire par relation.

Définition 6 *Clef étrangère d'une relation*

☞ Étant donné deux relations R et S, une **clef étrangère** de R est un attribut de R qui référence une clef primaire de S, c'est-à-dire qu'une valeur de la clef étrangère pour un nuplet de R doit être une valeur de clef primaire pour un nuplet de S.

☞ Dans le modèle relationnel, une clef étrangère doit être déclarée dans le schéma lors de la création de la base de données car elle constitue une **contrainte d'intégrité référentielle** garantissant la cohérence entre deux relations de la base.

Une relation doit comporter une clef primaire et une seule, mais elle peut contenir zéro, une ou plusieurs clef étrangères.

Exercice 3 : *Asie juin 2022 Partie 2*

On reprend la base de l'exercice 1.

1. Clés primaires/étrangères :

a. Donner la clé primaire de la relation `matchs`.

.....

.....

.....

b. La relation `matchs` a-t-elle une ou des clés étrangères ? Si oui quelles sont-elles ?

.....

.....

.....

.....

2. Par lecture et analyse des relations de la base de donnée

a. Déterminer le jour et la plage horaire du match entre Durand Belina et Caron Camilia.

.....

.....

.....

b. Déterminer le nom des deux joueurs qui sont les seuls à avoir joué dans le hangar.

.....

.....

.....

Exercice 4 *Métropole mai 2022 sujet 1*

Un musicien souhaite créer une base de données relationnelle contenant ses morceaux et interprètes préférés. Il crée une table `morceaux` qui contient entre autres les titres des morceaux et leur année de sortie :

id_morceau	titre	annee	id_interprete
1	Like a Rolling Stone	1965	1
2	Respect	1967	2
3	Imagine	1970	3
4	Hey Jude	1968	4
5	Smells Like Teen Spirit	1991	5
6	I Want To hold Your Hand	1963	4

Il crée la table `interpretes` qui contient les interprètes et leur pays d'origine :

id_interprete	nom	pays
1	Bob Dylan	États-Unis
2	Aretha Franklin	États-Unis
3	John Lennon	Angleterre
4	The Beatles	Angleterre
5	Nirvana	États-Unis

`id_morceau` de la table morceaux et `id_interprete` de la table `interpretes` sont des clés primaires. L'attribut `id_interprete` de la table morceaux fait directement référence à la clé primaire de la table `interpretes`.

1. Citer, en justifiant, la clé étrangère de la table `morceaux`.

.....

.....

.....

.....

2. Écrire un schéma relationnel des tables `interpretes` et `morceaux`.

.....

.....

.....

.....

3. Peut-on insérer dans la table `interpretes` le nuplet `(1, 'Trust', 'France')` ?

.....

.....

.....

.....

2.5 Contraintes d'intégrité

Propriété 2 ***Contraintes d'intégrité***

Lors de la création d'une base de données relationnelle, on doit déclarer son schéma en précisant pour chaque relation les domaines de ses attributs, la ou les clefs primaires et les éventuelles clefs étrangères. On définit ainsi des contraintes qui seront vérifiées par le SGBD lors de chaque transaction sur la base :

- *Contrainte de domaine* : la valeur d'un **attribut** doit toujours appartenir au **domaine** de cet attribut ;
- *Contrainte d'unicité (ou de relation)* : une valeur de **clef** ne peut apparaître qu'une fois dans une **relation** ;
- *Contrainte d'intégrité référentielle* :
  - la valeur d'une **clef étrangère** doit toujours être également une des valeurs de la clef référencée ;
  - une clef étrangère ne peut être une valeur qui n'est pas clé primaire de la relation à laquelle on se réfère ;
  - une valeur de clef primaire ne peut être changée dans une relation si le nuplet correspondant possède des nuplets liés par clef étrangère dans une autre relation ;
  - un nuplet de la relation primaire ne peut être effacé s'il possède des nuplets liés dans une autre relation de la base par une clef étrangère.

Les langages comme SQL permettent de définir lors de la création d'une base de données des contraintes utilisateurs sur les valeurs acceptables : par exemple l'âge en années d'un être humain doit appartenir au domaine `INT` *mais plus précisément à l'intervalle* $[0 ; 150]$.

Exercice 5 : *Asie juin 2022 Partie 3*

On considère que les relations de la base de données de l'exercice 2 contiennent uniquement les valeurs représentées page 8.

1. Peut-on supprimer de la table `creneaux`, le créneau dont la valeur de l'attribut `id_creneau` est 12 ? et celui dont l'attribut `id_creneau` a pour valeur 7 ?

   .....

   .....

   .....

   .....

   .....

2. Peut-on insérer dans la table `matchs` le nuplet `(4, "2020-08-09", 8, 3, 4, 3)` ? Sinon, quelle valeur d'attribut suffirait-il de modifier ?

   .....

   .....

   .....

   .....

3. Peut-on supprimer de la table `matchs` n'importe quel match ?

   .....

   .....

   .....

   .....

4. Peut-on insérer dans la table `matchs` le nuplet `(7, "2020-08-09", 8, 5, 4, 3)` ?

   .....

   .....

   .....

   .....

5. Peut-on changer la valeur de l'attribut `id_terrain` en 4 dans le nuplet de la table `matchs` dont la valeur de d'`id_match` est 5 ?

   .....

   .....

   .....

   .....