CENTRE NATIONAL D'ORIENTATION ET DE PREPARATION AUX CONCOURS D'ENTREE DANS LES GRANDES ECOLES ET FACULTES DU CAMEROUN

Préparation au Concours d'Entrée en Troisième Année de l'ENSP et FGI

Support de Cours

BASE DE DONNÉES

Avec Intelligentsia Corporation, Il suffit d'y croire !!!

☎ 698 222 277 / 671 839 797
fb : Intelligentsia Corporation
email : contact@intelligentsia-corporation.com

" Vous n'êtes pas un passager sur le train de la vie, vous êtes l'ingénieur. "

-- Elly Roselle --

Instructions :

Il est recommandé à chaque étudiant de traiter les exercices de ce recueil (du moins ceux concernés par la séance) avant chaque séance car ***le temps ne joue pas en notre faveur.***

I. INTRODUCTION & OBJECTIFS

Une **donnée** est la représentation d'un élément d'information, tel qu'un chiffre ou un fait, codé dans un format permettant son stockage et son traitement par ordinateur.

Une **information** est une donnée ou ensemble de données qui a ou été interprétée.

Une **base de données** (BD) peut être vue comme un ensemble structuré de données enregistrées sur des supports accessibles par des ordinateurs, représentant les informations du monde réel et pouvant être interrogées par différents utilisateurs.

C'est une structure assez complexe qui peut être définie à l'aide des mots clés : information, structurée, monde réel, données ou objets.

On en distingue plusieurs types :

- BD relationnelle : axée sur l'algèbre relationnelle,
- BD géographique : stockage des cartes géographiques,
- BD chimiques : stockage des molécules stables,
- etc.

Dans le cadre de ce cours, nous étudierons les BD relationnelles.

Une **BD relationnelle** (BDr) est une BD structurée suivant le principe de l'algèbre relationnelle (concept mathématique basée sur la théorie des ensembles et les opérations sur la relation). Les relations ici sont des tables. Une BDr est mise en jeux au moyen des systèmes de gestion des bases de données relationnelle (SGBDr).

Le principe (BDr) étant de stocker une grande quantité d'information et d'y naviguer à l'aide d'un SGBDr afin d'obtenir les informations voulues en un temps raisonnables.

Les objectifs majeurs d'un SGBD sont :

✓ Garantir l'intégrité des données : limiter l'altération (due aux pannes et aux erreurs) et l'incohérence des données (données en contradiction avec d'autre. **Ex :** Un âge négatif, deux adresses différentes pour une même personne, etc.

✓ La distinction entre données et traitements : les données existent indépendamment des traitements qu'on leur applique. Ainsi, on a d'un côté les données et leur modèle présentant une vision unifiée et de l'autre les traitements rationnalisés (les traitements se ramènent essentiellement à ajouter, retirer, modifier et consulter les données.).

✓ Performance et optimisation : Une BD doit fournir des performances acceptables par l'utilisateur. C'est la problématique de l'optimisation.

La conception d'une base de données relationnelle remplissant ces objectifs passe par plusieurs étapes qui sont :

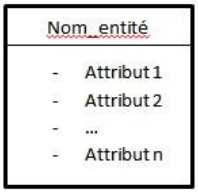
- Le modèle conceptuel de données (**MCD**)
- Le modèle logique de données (**MLD**)
- La normalisation des relations (optionnelle)
- Le modèle physique de données (**MPD**)

II. LE MODELE CONCEPTUEL DE DONNEES

Il part d'un domaine réel qu'il matérialise ou représente par un ensemble d'objets. Le modèle conceptuelle le plus utilisé est le modèle entité association (E/A) car il est facile pour migrer au modèle logique. Il repose sur les quatre éléments suivants : entité, association, attributs (ou propriété) et cardinalité.

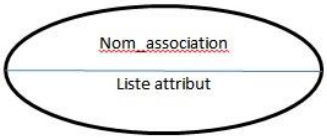
- Une **entité** est la représentation d'un élément matériel ou immatériel ayant un rôle dans le système que l'on désire décrire (**Ex :** Etudiant, voiture, filière…).

  On la représente par :

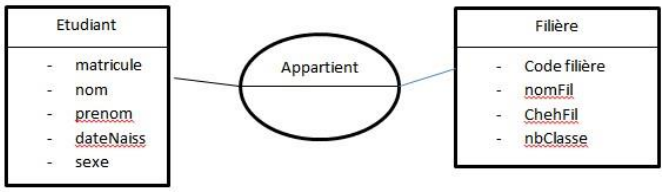


- Un **attribut** est une donnée élémentaire permettant de décrire une entité ou une association. Il prend généralement une valeur (**Ex :** Dans l'entité précédente, on peut avoir : sexe = 'féminin', matricule = '12P012', …).

- Une **association** est un lien sémantique entre une ou plusieurs entités. On la représente par :

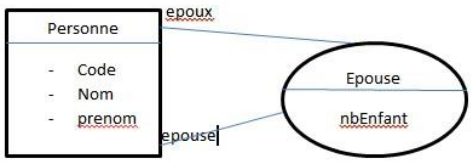


Ex : La relation un étudiant appartient à une filière est représenté par :



Une classe d'association ou de relation contient donc toutes les relations de même type. On distingue :

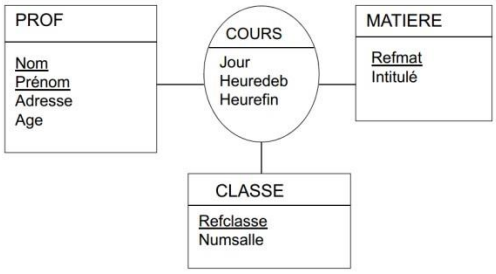
- Les **associations réflexibles** qui relient la même entité.



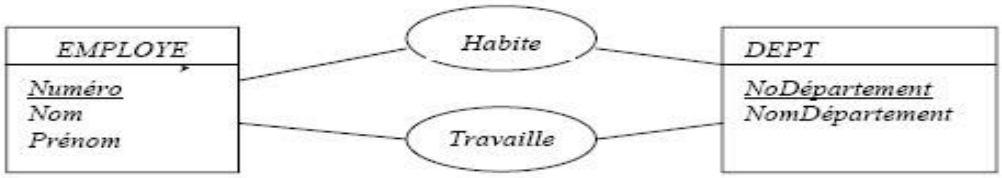
- Les **associations binaires** qui relient deux entités : c'est les plus répandues. **Ex :** Relation entre étudiant et filière.

- Les **associations n-aire** qui relient n entités :

  Ex : pour n = 3



Il peut exister plusieurs associations entre deux entités, chacune représentant une réalité différente :



- Un **identifiant** est un attribut ou ensemble d'attributs permettant de désigner une et une seule entité. La définition originale est la suivante: L'identifiant est une propriété particulière d'un objet telle qu'il n'existe pas deux occurrences de cet objet pour lesquelles cette propriété pourrait prendre une même valeur**.** Dans un modèle E/A les attributs formant l'identifiant sont soulignés. Les associations ne possèdent pas d'identifiant.

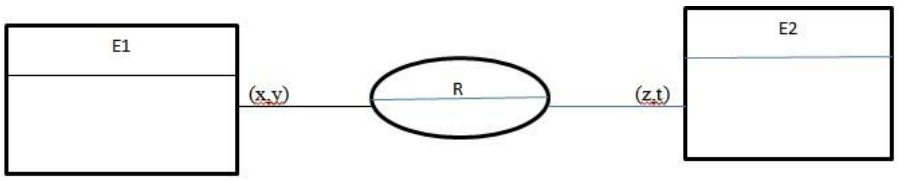
- Une **occurrence** : d'après la définition de l'identifiant, la connaissance de sa valeur détermine de manière unique les valeurs des autres attributs. L'ensemble de ses valeurs est appelé occurrence.

Ex : (14P001, KOUOGANG KAMDEM, Willy Hermann, 01/01/2018, masculin) est une occurrence de l'entité Etudiant et il n'existera plus une autre occurrence ayant pour matricule 14P001 et n'ayant pas de telles valeurs pour le nom, prenom, dateNaiss et sexe ;

- Les **cardinalités** : permettent de caractériser le lien qui existe entre une entité et la relation à laquelle elle est reliée. La cardinalité d'une relation est composée d'un couple comportant une borne maximale et une borne minimale, formant un intervalle dans lequel le nombre d'occurrence d'une entité peut prendre sa valeur :
  - La borne minimale (généralement 0 ou 1) décrit le nombre minimum de fois qu'une entité peut participer à une relation ;
  - La borne maximale (généralement 1 ou n) décrit le nombre maximum de fois qu'une entité peut participer à une relation ;

Une cardinalité placée entre une entité E et une association A représente le nombre minimal et maximal d'occurrences de E dans A. on distingue les cardinalités suivantes:

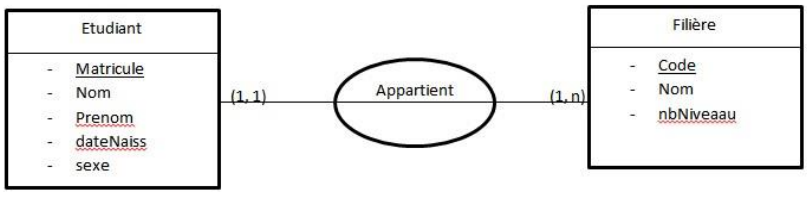
- (0,1) : entre une entité E et une association A signifie que chaque occurrence de E peut soit ne pas être relié à A soit l'être de manière unique.
- (1,1) : entre une entité E et une association A signifie que chaque occurrence de E est relié exactement une seule fois à A ;
- (0,n) : entre une entité E et une association A signifie que chaque occurrence de E peut soit ne pas être relié à A ou l'être à plusieurs reprises.
- (1,n) : entre une entité E et une association A signifie que chaque occurrence de E est relié à A une ou plusieurs fois.



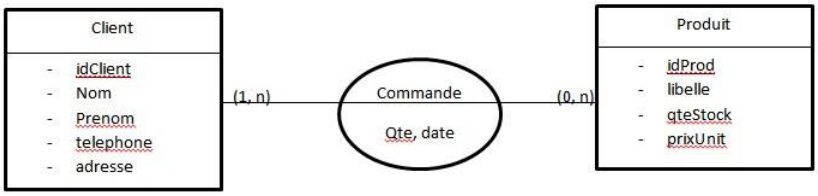
Un tel modèle se lit : un E1 R x ou y E2 et un E2 R z ou t E1.

Exemples :

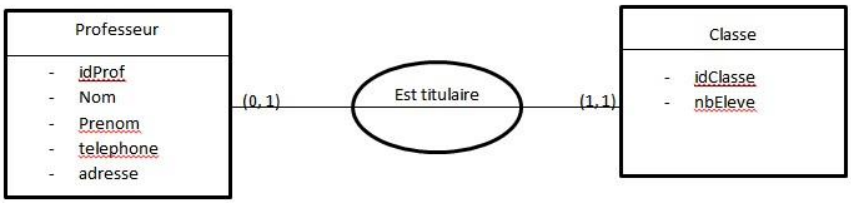
1) Un étudiant appartient à une et une seule filière et une filière contient un ou plusieurs étudiants.



2) Un client commande un ou plusieurs produits et un produit peut être commandé par un ou plusieurs clients ou ne pas être commandé.

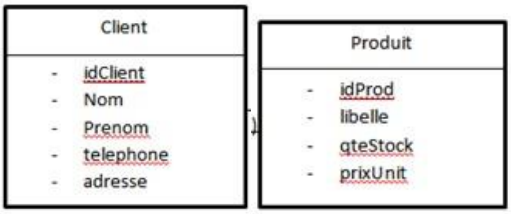


3) Une classe a un et un seul prof titulaire et un prof est soit titulaire dans une seule classe soit dans aucune.

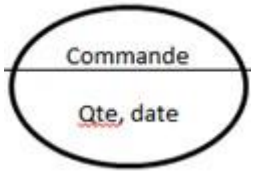


Résumé: le schéma Entité-Association

- **Entité** : représentation d'un ensemble d'objets abstraits ou concrets, caractérisée par une liste d'attributs. Un ou plusieurs attributs jouent le rôle de clé.

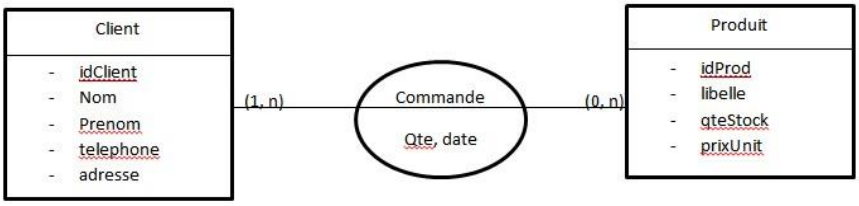


- **Association :** permet de décrire les liens "sémantiques" entre des entités, peut être caractérisée par des attributs mais ne possède pas de clé primaire ou identifiant.



- **Attribut** (mono-valué) : décrit une propriété attachée soit à une entité, soit à une association. Prend ses valeurs dans un domaine simple (chaîne, entier, réel…).

- **Cardinalité :** (0, 1), (1, 1), (0, n), (1, n) qui indique le nombre de fois qu'une entité entre dans une relation.



Le MCD ci-dessous signifie qu'un client commande 1 ou plusieurs produits et qu'un produit peut être commandé par un ou plusieurs clients ou ne pas être commandé.

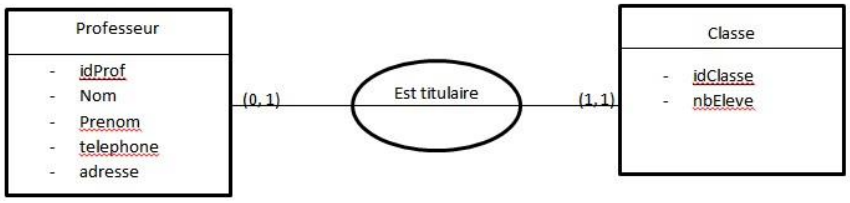
III. LE MODELE LOGIQUE (OU MODELE RELATIONNEL) DE DONNEE (MLD)

Le MLD est une suite normale du processus de conception d'une base de données. Son but est de nous rapprocher au plus près du modèle physique.

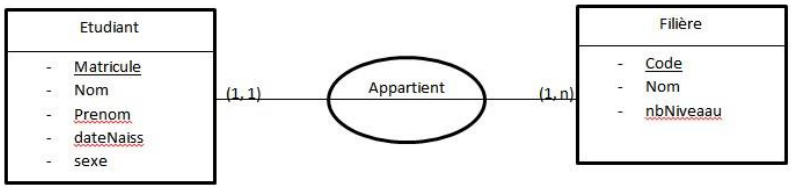
Pour cela, nous partons du MCD et nous lui enlevons les associations, mais pas n'importe comment, il faut en effet respecter certaines règles.

On distingue trois types d'associations :

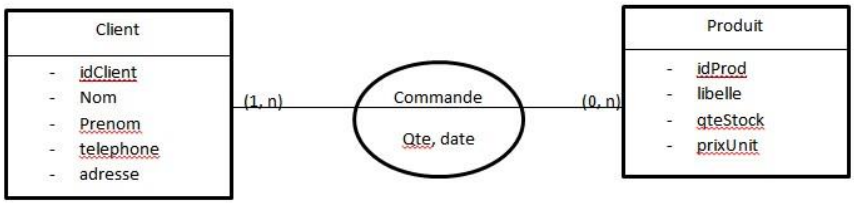
- Les **associations 1 à 1** (ou one to one en anglais) : ce sont des associations dans lesquelles les cardinalités des entités sont (0, 1) ou (1, 1) de chaque côté.



- Les associations **un à plusieurs** (one to many) : ce sont les associations dans lesquelles les cardinalités des entité sont (0,1) ou (1,1) d'une part et (0, n) ou (1,n) d'autre part.



- Les associations **plusieurs à plusieurs** (many to many) : ce sont les associations dans lesquelles les cardinalités des entités sont (0, n) ou (1, n) des deux côtés.



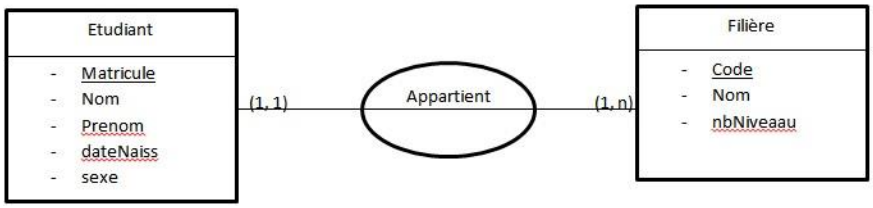
Les règles de passage du MCD au MLD sont les suivantes :

1) Transformer toute entité E en une relation RE dont les attributs sont les attributs de E et la clé primaire de RE est un des identifiants de E.

Ex : pour l'entité Etudiant, on aura : Etudiant (matricule, nom, prénom, dateNaiss, sexe).

2) Pour toute association de type un à plusieurs, l'entité dont la cardinalité maximale est à 1 recevra l'identifiant de l'entité dont la cardinalité maximale est à n ainsi que tous les attributs de l'association.

Ex : cas de l'association entre un Etudiant et une filière.

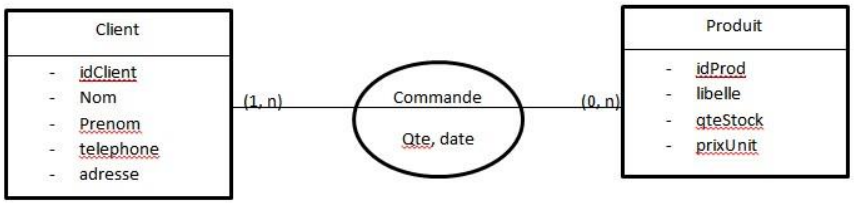


Etudiant (matricule, nom, prénom, dateNaiss, sexe, #code)

Filière (code, nom, nbNiveau)

3) Pour toute association de type plusieurs à plusieurs, l'association se transforme en une relation dont la clé primaire est formée par les identifiant des entités mise en relation.

Ex : association entre client et produit :

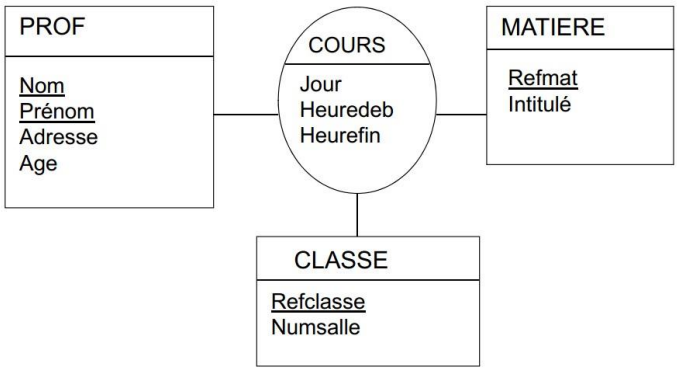


Client (idClient, nom, prenom, telephone, adresse)

Produit (idProd, libelle, qteStock, prixUnit)

Commande (#idClient, #idProd, qte, date)

4) Toute association n-aire (n > 2) devient une relation dont la clé primaire est formée par les identifiants des entités mises en relation.



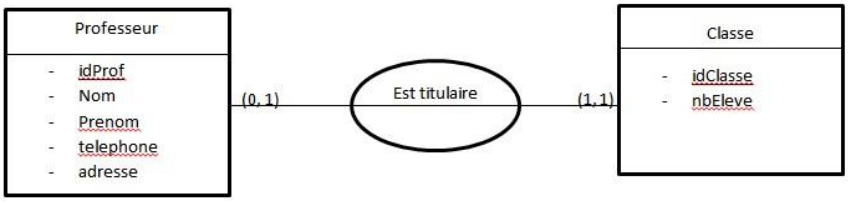
Prof (nom, prenom, adresse, age)

Matiere (refMat, intitulé)

Classe (refClasse, numSalle)

Cours (#nom, #prenom, #refMat, #refClasse, jour, heureDeb, heureFin)

5) pour toute association de type un à un, l'identifiant d'une entité migre dans l'autre entité, ainsi que tous les attribut de l'association.



Professeur (idProf, nom, prenom, telephone, adresse)

Classe (idClasse, nbEleve, #idProf)

IV. DEPENDANCES FONCTIONNELES & NORMALISATION (DF & NF)

1. Dépendances fonctionnelles

Un attribut (ou une liste d'attributs) Y **dépend fonctionnellement** d'un attribut (ou d'une liste d'attributs) X dans une relation R, si étant donnée une valeur de X, il ne lui est associé qu'une seule valeur de Y dans toute instance de R.

On note X ⟶ Y et on lit X détermine Y ou encore Y dépend de X. X ⟶ Y est appelé **dépendance fonctionnelle**.

Exemples :

Soit la relation : Commande (idClient, nomClient, prenomClient, idProduit, libelle, prixU, qteCmd)

- idProd ⟶ libellé, prixU (idProd determine libellé et prix)
- idClient⟶ nomClient, prenomClient
- idProd, idClient ⟶ qteCmd (idProd et idClient determinent qteCmd).

On a les propriétés suivantes sur les DF :

Soit U la liste des attributs d'une relation R; W, X, Y et Z des attributs (ou ensembles d'attributs) de R tels que : W, X, Y et Z soient inclus dans U.

- Réflexivité : si Y est inclus dans X alors X ⟶ Y

  Ex : nomClient, prenomClient ⟶ nomClient

- Augmentation : si X ⟶ Y alors X, Z ⟶ Y, Z

  Ex : si idProd ⟶ libellé alors idProd, nomClient ⟶ libellé, nomClient

- Transitivité : si X ⟶ Y et Y ⟶ Z alors X ⟶ Z

  Ex : titreFilm ⟶ realisateur et realisateur ⟶ categorie

  alors titreFilm ⟶ categorie

- Union : si X ⟶ Y et X ⟶ Z alors X ⟶ Y, Z

  Ex : idProd ⟶ libellé et idProd ⟶ prixU alors idProd ⟶ libellé, prixU

- Pseudo-transitivité : si X ⟶ Y et Y, W ⟶ Z alors X, W ⟶ Z

Ex : idProd ⟶ libellé et libellé, idClient ⟶ nomClient

alors idProd, idClient ⟶ nomClient

- Décomposition : si X ⟶ Y, Z alors X ⟶ Y et X ⟶ Z

Ex : idProd ⟶ libellé, prixU alors idProd ⟶ libellé et idProd ⟶ prixU

Soit U la liste des attributs d'une relation R, on appelle clé primaire de R un attribut (ou ensemble minimum d'attributs) X de R tel que X ⟶ U\X. En d'autres termes c'est l'ensemble minimal d'attributs de R qui permettent d'obtenir tous les autres attributs.

Ex : Dans la table Commande précédente, on a les DF suivantes :

idProd ⟶ libellé, prixU ;

idClient ⟶ nomClient, prenomClient ;

idProd, idClient ⟶ qteCmd.

Avec les attributs idprod et idClient on determine tous les autres. Donc la clé primaire de Commande est formée par idProd et idClient.

2. Normalisation

La normalisation est une technique utilisée pour corriger les défauts de conception. En effet, à l'issu d'une conception (MCD), on peut obtenir un MLD qui présente plusieurs anomalies notamment les anomalies d'insertion, de suppression et de modification. Soit la relation suivante qui représente les produits commandés par des clients :

idClient	nomClient	prenomClient	idProduit	libelle	prixU	qteCmd
008	kamgang	hubert	012	tangui	400	3
001	tanko	gislain	005	laptop	320000	1
006	atangana	vincent	053	plat	1500	10
008	kamgang	hubert	098	television	55000	2
045	Balango	thomas	012	tangui	400	10
023	zanga	venant	053	plat	1500	50
001	tanko	gislain	012	tangui	400	45
008	kamgang	hubert	040	tables	3500	10

- Chaque fois qu'un client commande un produit, il faut remplir son identifiant, son nom et son prénom, pourtant ayant son code, on peut retrouver son nom et son prénom. Il y a redondance des informations (nom et prénom du client). C'est une anomalie d'insertion.
- S'il fallait modifier le nom ou le prénom d'un client, il faudra le faire autant de fois qu'il existe dans la table c'est-à-dire autant de fois qu'il a acheté un produit sinon, il y aurait une incohérence des données.
- Si l'on venait à supprimer le client « kamgang hubert », on supprimera également le produit télévision et perdrait ainsi les informations le concernant. C'est une anomalie de suppression.

Pour être parfaites, les relations doivent respecter certaines règles. Ces règles se nomment : les formes normales. Elles utilisent les dépendances fonctionnelles entre les attributs.

- Première forme normale (1FN ou 1NF) :

Une relation R est en 1FN si elle admet une clé primaire et si tous ses attributs sont mono-valués.

Ex : Personne (idPers, nomPers, telPers, metier, nomEnfant) ayant les valeurs suivantes :

idPers	nomPers	telPers	metier	nomEnfant
0012	Kamgang hubert	94586231	enseignant	-tanko gislain -zanga venant -dezo audrès
0050	Balango thomas	95124832	medecin	-songa christian
0018	Fomanko muriel	75489623	chauffeur	-kenne anne -Atangana vincent

Cette relation Personne n'est pas en 1FN car l'attribut nomEnfant n'est pas mono-valué. Pour rendre une relation R en 1FN on supprime de R l'attribut multi-valué, on crée une nouvelle relation R1 ayant pour attribut la clé primaire de R et l'attribut multi-valué.

Personne (idPers, nomPers, telPers, metier)

Enfant (idEnf, idPers, nomEnfant)

- Deuxième forme normale (2FN ou 2NF) :

Une relation R est en 2FN si elle est en 1FN et si tout attribut non clé (n'appartenant pas à la clé primaire) dépend fonctionnellement de toute la clé primaire c'est-à-dire de tous les attributs la composant.

La relation Commande est en 1FN mais pas en 2FN car nomClient et prenomClient dépendent d'une partie de la clé (idClient) de même que libellé et prixU qui dépendent uniquement de idProd.

Pour rendre une relation R en 2FN, il faut la rendre en 1FN puis isoler tous les attributs qui dépendent d'une partie de la clé dans une nouvelle relation ainsi que les attributs de la clé dont ils dépendent. Dans le cas de la relation Commande, on aura :

Commande (#idClient, #idProd, qteCmd)

Client (idClient, nomClient, prenomClient)

Produit (idProd, libelle, prixU)

- Troisième forme normale (3FN ou 3NF) :

Une relation R est en 3FN si elle est en 2FN et si tous attribut non clé dépend directement de la clé (c'est-à-dire s'il n'y a pas de transitivité entre les attributs non clé de R).

Soit la relation : Avion (idAvion, nom, constructeur, type, capacité) avec les DF suivante :

idAvion ⟶ nom, constructeur, type, capacité

type ⟶ constructeur, capacité

Cette relation est en 2FN mais pas en 3FN car capacité et constructeur dépendent d'un attribut qui n'est pas la clé. Pour rendre une relation R en 3FN, on la mets en 2FN puis on isole les attributs qui dépendent d'un ou des attribut(s) non clé ainsi que ces attributs. Dans le cas de la relation Avion, on aura :

Avion (idAvion, nom, #type)

Modèle (type, constructeur, capacité)

NB : Il existe d'autres formes normales plus poussées que les trois ci-dessus, c'est le cas de BCNF (forme normale de Boyce-Codd), 4NF (quatrième forme normale), etc. Du fait de leur caractère plus théorique que pratique, elles ne seront pas étudiées dans ce cours.

V. ELEMENTS D'ALGEBRE RELATIONNELLE

L'**algèbre relationnelle** est une théorie mathématique proche de la théorie des ensembles qui définit des opérations qui peuvent être effectuées sur des relations.

Le **schéma** d'une relation est l'ensemble de ses attributs dans un ordre précis.

Un **tuple** est n-uplet de données dont chaque élément est une valeur d'un attribut dans une relation (ou table).

Ex : Soit la table Employé (idEmp, nomEmp, sexe, poste) :

idEmp	nomEmp	sexe	poste
001	Dupont	masculin	ingénieur
068	Amélie	féminin	commercial

(001, "Dupont", masculin, "ingénieur") et (068, "Amélie", féminin, "commercial") sont des tuples de Employé.

1. Opérateurs ensemblistes

- Union : R1 U R2

  Tous les tuples appartenant soit à R1, soit à R2, soit à R1 et à R2.

- Intersection : R1 ∩ R2

  Tous les tuples appartenant à R1et à R2.

- Différence : R1 - R2

  Tous les tuples appartenant à R1 et n'appartenant pas à R2.

  NB : Pour les trois opérateurs précédents, R1 et R2 doivent avoir le même schéma.

- Produit cartésien : **R1 x R2**

  Tous les tuples formés d'un tuple de R1 et d'un tuple de R2.

  Soient R1 (A1, A2, …, An) et R2 (B1, B2, …, Bp) :

  R1 x R2 est défini par R (A1, A2, …, An, B1, B2, …, Bp) tel que :

  (a1, a2, …, an, b1, b2, …, bp) ∈ R ssi (a1, a2, …, an) ∈ R1 et (b1, b2, …, bp) ∈ R2

R1

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3

R2

X	Y
x1	y1
x2	y2

PRODUIT CARTESIEN

R1XR2

A	B	C	X	Y
a1	b1	c1	x1	y1
a2	b2	c2	x1	y1
a3	b3	c3	x1	y1
a1	b1	c1	x2	y2
a2	b2	c2	x2	y2
a3	b3	c3	x2	y2

commutatif: [R1 x R2] = [R2 x R1]

associatif: [(R1 x R2) x R3] = [R2 x (R1 x R3)]

2. Opérateurs relationnels

- Sélection : $\sigma_C(R)$

  Tous les tuples de R qui vérifient la condition **C**.

  Ex : $\sigma_{adresse="Paris"}(Client)$

- projection : $\pi_A(R)$

  Tous les tuples de R en considérant uniquement les attributs qui sont dans **A**.

  Ex : $\pi_{nom,\ adresse}(Client)$

- Rebaptisation ou renommage : $\rho_{A1:B1,\ \ldots,\ An:Bn}(R)$

  Tous les tuples de R en renommant les colonnes **Bi** en **Ai**.

  Ex : $\rho_{nom:nomClient,\ id:idClient}(Client)$

- Jointure : $R1 \underset{C}{\bowtie} R2$

  Tous les tuples de R1 x R2 qui vérifient la condition **C**.

  Ex : $Client \underset{dateNais > dateCommande}{\bowtie} Commande$

- Jointure naturelle : $R1 \quad R2$

R1 et R2 doivent avoir un attribut (ou une liste d'attributs) **A** en commun.

$$R1 \triangleright\!\!\bowtie\!\!\triangleleft R2 \equiv R1 \underset{R1.A = R2.A}{\triangleright\!\!\bowtie\!\!\triangleleft} R2$$

- Division : **R1 / R2** ou **R1 ÷ R2**

  Tous les attributs de R2 doivent être inclus dans R1.

  Supposons R1 (A1, A2, …, An, B1, B2, …, Bp) et R2 (B2, …, Bp)

  Cette division retourne tous les tuples (a1, a2, …, an) de $\pi_{A1,A2,\ldots,An}(R1)$ tels que pour tout tuple (b1, b2, …, bn) de R2, (a1, a2, …, an, b1, b2, …, bp) ∈ R1.

  Exemple :

  Client (idClient, nomClient, adresse)
  Produit (refProduit, nomProduit)
  Vente (idVente, #idClient, #refProduit, date)

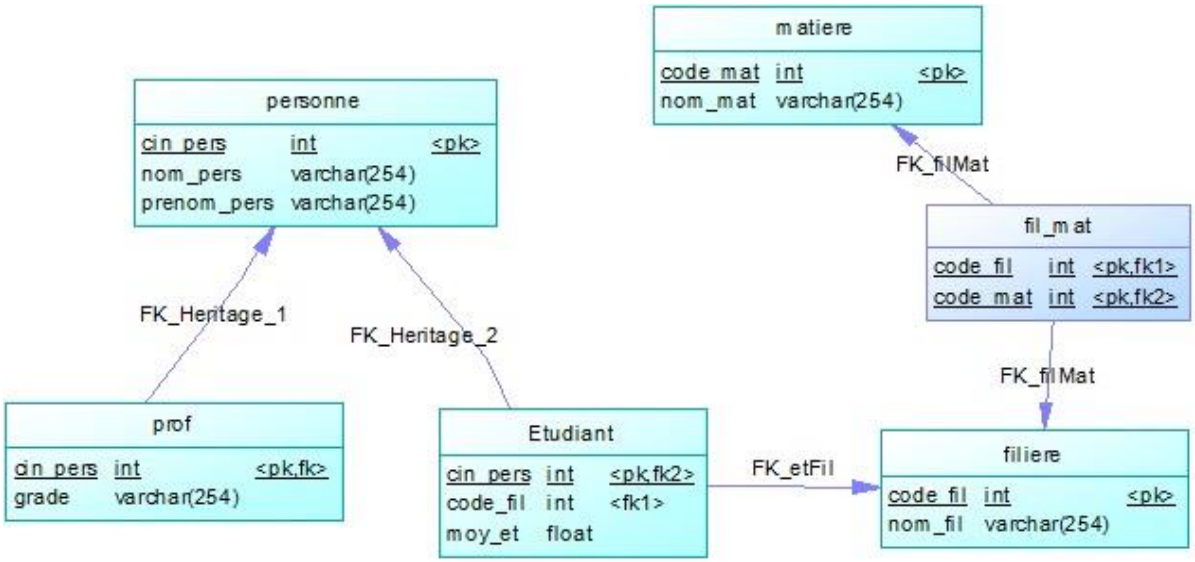
  Q : Quelles sont les références des produits achetés par tous les clients ?

  R : $\pi_{refProduit,\ idClient}(Vente) \div \pi_{idClient}(Client)$

VI. MODELE PHYSIQUE DE DONNEES (MPD)

Le MPD utilise le MLD. Il représente toutes les relations du MLD sous forme d'entité en précisant les relations de références entres les attributs de ces relations et le type de chaque attribut.

Exemple :



VII. COMPLEMENT : LES REQUETES SQL

Les **requêtes SQL** (Structured Query Language) sont l'application de l'algèbre relationnelle à un SGBDr.

Il s'agit ici d'interroger une base de données réelle pour avoir une information précise.

La structure de base d'une requête SQL est la suivante :

```
SELECT [ DISTINCT ] A1, A2, ..., An        --colonnes
FROM R                                     --relation
[ WHERE F ]                                --assertion
[ GROUP BY A                               --regroupement
[ HAVING H ] ]                             --assertion
[ ORDER BY T ]                             --tri
;
```

NB: Tout ce qui est entre crochet n'apparait pas dans toutes les requêtes, ce sont des blocs optionnels.

Explication des mots clés :

➢ SELECT :

C'est une projection (en algèbre relationnelle), il spécifie l'ensemble des colonnes qui doivent apparaitre dans le résultat. Possibilité de prendre toutes les colonnes en utilisant '*' (SELECT * FROM R).
Il est possible de rebaptiser une colonne en utilisant le mot clé **AS** (SELECT id AS idEmp FROM employe) ou d'appliquer directement une opération sur une colonne (SELECT 2018 – anneeNais AS age FROM etudiant).

➢ DISTINCT :

Il est optionnel, il permet d'éliminer les doublons dans le résultat.

➢ FROM :

Il permet de spécifier la (les) table(s) dans laquelle (lesquelles) la projection sera effectuée.

➢ **WHERE :**

C'est une sélection (en algèbre relationnelle), il permet de poser une assertion qui doit être vérifiée par les lignes de la table solution (SELECT * FROM etudiant WHERE 2018 – anneeNais >= 18).

➢ **GROUP BY :**

Il permet de créer des classes distinctes dans les tuples d'une relation suivant un critère et d'appliquer un opérateur d'agrégation à chacune de ces classes (SELECT * FROM etudiant GROUP BY anneeNais).

➢ **HAVING :**

Il permet d'appliquer un critère de sélection aux lignes d'une partition créée par GROUP BY (SELECT * FROM etudiant GROUP BY anneeNais HAVING anneeNais < 2000).

➢ **ORDER BY :**

Il permet de fournir un résultat trié suivant certaines colonnes (SELECT * FROM etudiant ORDER BY anneeNais).
Il est possible de renseigner l'ordre de tri avec **ASC** pour un tri par ordre croissant et **DESC** pour un tri par ordre décroissant (à noter que par défaut c'est ASC).
Ex : SELECT * FROM article ORDER BY datePublication DESC

Opérations sur les lignes :

➢ **MIN()** et **MAX()** :

Ils retournent respectivement la plus petite et la plus grande valeur d'une colonne (SELECT MAX(datePublication) FROM article).

➢ **SUM()** :

Permet de sommer une colonne contenant des valeurs numériques (SELECT SUM(prix) FROM produit).

➢ **AVG()** :

Permet d'avoir la moyenne d'une colonne contenant des valeurs numériques (SELECT AVG(note) FROM etudiant).

➢ **COUNT()** :

Retourne le nombre de valeurs dans une colonne (SELECT COUNT(idVente) FROM vente).

Opérateurs binaires entre les tables :

- **UNION, INTERSECT, EXCEPT, CROSS JOIN:**

Permettent respectivement l'union, l'intersection, la différence et le produit cartésien entre de deux tables (SELECT * FROM etudiant UNION enseignant).

- **JOIN … ON:**

Permet de faire la jointure entre deux tables (SELECT * FROM client JOIN commande ON dateNais > dateCommande).

- **NATURAL JOIN :**

Jointure naturelle…

Remarques importantes :

- L'opérateur de division est un concept purement théorique et n'existe donc pas en SQL. Toutefois il possible d'aboutir à ce résultat en utilisant les opérateurs existants.

- Il est possible d'imbriquer une requête dans une autre.

Ex : SELECT nom FROM client WHERE id = (SELECT idClient FROM vente WHERE idVente = MAX(idVente))

- Il existe un opérateur **IN** qui permet de verifier si un élément appartient à un ensemble.

Exple1 : SELECT nom FROM client WHERE id IN (SELECT idClient FROM vente WHERE date >= '2018-01-30')

Exple2 : SELECT nom FROM client WHERE profession IN ('Ingénieur', 'Médecin', 'Enseignant')