Université Virtuelle du Sénégal

Pôle des Sciences, Technologies et Numérique

Sciences Informatiques et Mathématiques de la Cybersécurité

Licence 2

SIMAC2312 - Base de données

Juillet 2022

©Dr Guy MBATCHOU

+221 76 600 79 40

guy.mbatchou@uvs.edu.sn

A propos

1 - Organisation pédagogique

- Pôle : Sciences, Technologies et Numérique
- Domaine : Sciences et Technologies
- Mention : Sciences Informatiques et Mathématiques de la Cybersécurité
- Spécialité :
- Cycle : Licence
- Niveau : 2
- Semestre : 3

2 - Unité d'enseignement

- **Code** : LSIMAC231
- **Intitulé** : Systèmes d'Information et Audit
- **Crédit** : 11

3 - Elément constitutif

- **Code** : LSIMAC2312
- **Intitulé** : Bases de données
- **Poids de l'EC au sein de l'UE** : 27,27%
- **Statut :** Obligatoire
- **Nombre de séquences** : 6
- **Nombre d'objectifs généraux** : 3
- **Nombre d'objectifs spécifiques** : 17
- **Durée** : 3 semaines
- **Version :** 1.0
- **Date de création ou de mise à jour :** Juillet 2022

4 - Charge de travail

- **Volume horaire total :** 60 heures
- **Apprentissage supervisé : 36** heures
- **Travail Personnel de l'Etudiant :** 24 heures

- **Tutorat** : 20 heures
- **Durée apprentissage par semaine** :

5 - Modalités

- **Spatiale** : A distance
- **Temporelle** : Mixte (synchrone et asynchrone)
- **Collaborative** : Travail individuel

6 - Objectifs généraux

- Comprendre la nécessité d'une base de données
- Savoir concevoir une base de données relationnelle
- Savoir implémenter une base de données dans un Système de Gestion de Base de Données

7 - Mots-clés

- Base de données
- Entité
- Relation
- Association
- Modélisation des données
- Schéma de base de données
- Instance de base de données
- Système de Gestion de Base de Données

8 - Prérequis

- Se connecter à un ordinateur sous Microsoft Windows
- Sécuriser sa session de travail, redémarrer et éteindre un ordinateur
- Organiser les données dans un ordinateur

9 - Test d'entrée

Voir fichier joint des travaux dirigés et pratiques

10 - Références Bibliographiques

[1] Jean-Luc Hainaut ; Bases de données : Concepts, utilisation et développement ; Dunod ; 2018 ; https://international.scholarvox.com/reader/docid/88863929/page/1

[2] Christian Soutou, Frédéric Brouard ; Modélisation des bases de données : UML et les modèles entité-association ; Eyrolles ; 4ème édition ; 2017 ; https://international.scholarvox.com/reader/docid/88842073/page/1

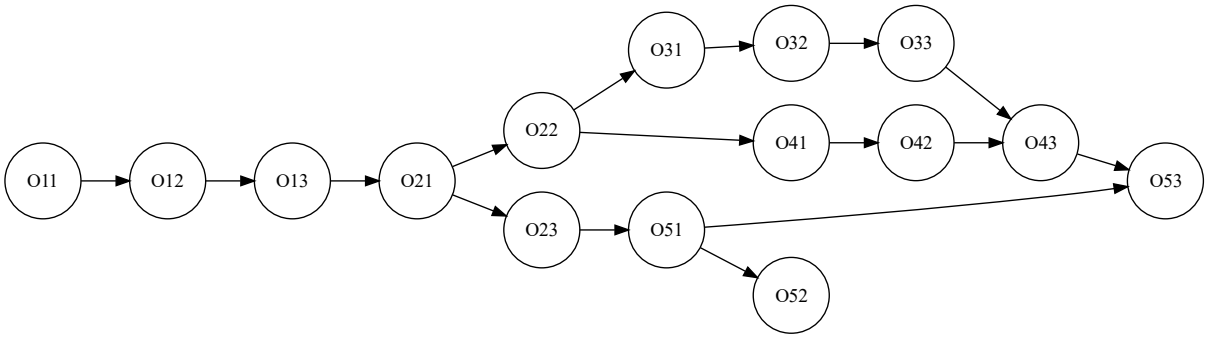
[3] Le langage SQL ; https://sql.sh/

11 - Déroulement de l'apprentissage

Séquences pédagogiques			Objectifs pédagogiques			
N°	Titre	Durée	Code	Intitulé	(%) Seuil Validation	Durée (Heure)
1	Des données à une Base de Données	0.5 jour	11	Définir la notion de donnée	60	4
			12	Montrer l'intérêt d'un fichier	60	4
			13	Spécifier l'intérêt de passer d'un fichier à une base de données	75	4
2	Généralités sur les bases de données	2 jours	21	Expliquer une base de données	50	12
			22	Décrire les étapes de la conception d'une base de données	80	12
			23	Décrire un Système de Gestion de Base de Données (SGBD)	50	24
3	Modélisation conceptuelle des données : Modèle Entité-Association	7 jours	31	Recenser les données élémentaires du monde à modéliser pour constituer un dictionnaire de données	60	48
			32	Constituer les entités à partir des données élémentaires	75	48
			33	Mettre les entités en relation pour constituer le Modèle Entité-Association	75	72
4	Modélisation logique des donnés : Modèle Relationnel	3 jours	41	Décrire les principes du modèle relationnel	60	6
			42	Décrire la notion de relation	50	6
			43	Passer d'un Modèle Entité-Association à un Modèle Relationnel de données	80	30
			44	Normaliser un modèle relationnel	80	30
5	Interrogation d'une base de données : algèbre relationnelle	3 jours	51	Ecrire des requêtes d'interrogation d'une base de données relationnelles	60	72
6	Implémentation d'une base de données : Modèle physique	5,5 jours	61	Décrire les principes de la gestion informatisée d'une base de données	60	6
			62	Implémenter une base de données	75	48

Séquences pédagogiques			Objectifs pédagogiques			
N°	Titre	Durée	Code	Intitulé	(%) Seuil Validation	Durée (Heure)
			63	Interroger une base de données en langage SQL	75	78

12 - Graphe de précédences entre les objectifs



13 - Consignes de travail

- Faites le Test d'entrée ; en principe vous devez avoir un score de 80% pour pouvoir continuer mais cette contrainte n'est pas prise en compte.
- Le cours est organisé par objectifs qui s'afficheront dès que vous aurez atteint l'objectif courant
- Dans chaque objectif, il faut d'abord lire le cours, ensuite la vidéo et enfin les exercices
- Le tout premier exercice de chaque objectif est un Test de connaissance auquel votre note doit être égale à au moins à 80%. Vous avez la possibilité de reprendre le test jusqu'à obtenir les 80% sinon vous ne pouvez pas progresser dans votre apprentissage
- Les autres exercices sont des exercices de production classiquement appelé Travaux Dirigés ou Travaux Pratiques. Vous devez déposer votre solution pour progresser
- Tous les exercices ne sont pas obligatoires et le système vous donnera progressivement les conditions d'accès à chaque exercice. Je vous recommande de faire le maximum !
- Vous aurez des séances de travail synchrone avec un tuteur pour des explications en direct

14 - Liste des figures

Figure 1 : Processus de conception d'une base de données 16
Figure 2 : Position du SGBD par rapport à la base de données 20
Figure 3 : Architecture d'un Système de Gestion de Base de Données 23
Figure 4 : Architecture à 3 niveaux d'ANSI/SPARC 25
Figure 5 : Formalisme d'une entité-type selon Peter Chen 34
Figure 6 : Exemple d'une entité-type selon Peter Chen 34

Figure 7 : Formalisme d'une entité-type selon la méthode MERISE 35
Figure 8 : Exemple d'une entité-type selon MERISE 35
Figure 9 : Occurrence de l'entité-type produit selon Peter Chen 35
Figure 10 : Occurrence de l'entité-type produit selon MERISE 35
Figure 11 : Exemple d'une entité avec identifiant selon Peter Chen 39
Figure 12 : Exemple d'une entité avec identifiant selon MERISE 39
Figure 13 : Liste des entités du site d'e-commerce représentées selon la méthode MERISE 40
Figure 14 : Formalisme d'une association-type selon Peter Chen 42
Figure 15 : Formalisme d'une association-type selon la méthode MERISE 42
Figure 16 : Exemple d'association déterminée à partir une dépendance fonctionnelle entre les identifiants de 2 entités et représentée selon Peter Chen 43
Figure 17 : Exemple d'association déterminée à partir une dépendance fonctionnelle entre les identifiants de 2 entités et représentée la méthode MERISE 43
Figure 18 : Exemple d'association déterminée à partir une dépendance fonctionnelle avec une source composée des identifiants primaires et représentée selon Peter Chen 44
Figure 19 : Exemple d'association déterminée à partir une dépendance fonctionnelle avec une source composée des identifiants primaires et représentée selon le modèle MERISE 44
Figure 20 : Exemple 1 de cardinalités représentées selon la méthode MERISE 45
Figure 21 : Exemple 1 de cardinalités représentées selon la Peter Chen 45
Figure 22 : Exemple 2 de cardinalités représentées selon la méthode MERISE 45
Figure 23 : Exemple 2 de cardinalités représentées selon Peter Chen 45
Figure 24 : Exemple d'association réflexive représentée selon Peter Chen 47
Figure 25 : Exemple d'association réflexive représentée selon la méthode MERISE ... 48
Figure 26 : Formalisme du lien de spécialisation ou généralisation 49
Figure 27 : Exemple de représentation des liens d'héritage et spécialisation 50
Figure 28 : Modèle Entité-Association du site d'e-commerce représenté selon la méthode MERISE 51
Figure 29 : Représentation par un graphe d'une relation entre 2 ensembles 54
Figure 30 : Représentation sous forme de tableau d'une relation entre 2 ensembles 55
Figure 31 : Passage d'une association « père-fils » au modèle relationnel 60
Figure 32 : Passage d'une association « Plusieurs à Plusieurs » au modèle relationnel. 61
Figure 33 : Passage d'une association « au plus un : (0,1) à (1,1) » au modèle relationnel 62
Figure 34 : Passage d'une association « un à un : (1,1) à (1,1) avec indépendance existentielle entre les entités » au modèle relationnel 62
Figure 35 : Passage d'une association « un à un : (1,1) à (1,1) avec dépendance existentielle entre les entités » au modèle relationnel 63
Figure 36 : Passage d'une association récursive ou réflexive au modèle relationnel 64
Figure 37 : Passage d'une spécialisation complète au modèle relationnel 64
Figure 38 : Passage d'une spécialisation complète au modèle relationnel 65
Figure 39 : Modèle relationnel de données du site d'e-commerce 66

Figure 40 : Exemple de relation qui n'est en 1FN 68
Figure 41 : Normalisation en 1FN de la Figure 40 68
Figure 42 : Contre-exemple typique d'une relation qui n'est pas en 2FN 68
Figure 43 : Exemple de relation qui n'est pas en 2FN 69
Figure 44 : Normalisation en 2FN de la Figure 42 69
Figure 45 : Contre-exemple typique d'une relation qui n'est pas en 3FN 69
Figure 46 : Exemple de relation qui n'est pas en 3FN 70
Figure 47 : Normalisation en 3FN de la Figure 42 70
Figure 48 : Contre-exemple typique d'une relation qui n'est pas en FNBC 70
Figure 49 : Exemple de relation qui n'est en FNBC 70
Figure 50 : Normalisation en FNBC de la Figure 49 71
Figure 51 : Représentation graphique de l'opérateur de sélection ou restriction 74
Figure 52 : Représentation graphique de la requête de sélection des produits à approvisionner en urgence 74
Figure 53 : Représentation graphique de l'opérateur de projection 74
Figure 54 : Représentation graphique de la requête de la projection de la relation Client sur les attributs : nom, prénom, téléphone et adresse électronique 75
Figure 55 : Représentation graphique de l'opérateur d'union 76
Figure 56 : Représentation graphique de l'opérateur d'intersection 76
Figure 57 : Représentation graphique de l'opérateur de différence 77
Figure 58 : Représentation graphique de l'opérateur du produit cartésien 77
Figure 59 : Représentation graphique de l'opérateur de la division 78
Figure 60 : Représentation graphique de l'opérateur de la jointure naturelle 78
Figure 61 : Représentation graphique de l'opérateur de la théta-jointure ou inéqui-jointure 79
Figure 62 : Représentation du Modèle Physique de Données du site e-Commerce réalisé avec le logiciel Microsoft Access 87
Figure 63 : Syntaxe SQL pour l'insertion des données dans une table 87
Figure 64 : Exemples d'insertion des données dans la table Fournisseur 87
Figure 65 : Exemple d'insertion de plusieurs lignes ou tuples dans la table Fournisseur 88
Figure 66 : Résultat de l'insertion multiple dans la table Fournisseur 88
Figure 67 : Syntaxe SQL pour la modification des données d'une table 88
Figure 68 : Exemple de modification des données de table Fournisseur 88
Figure 69 : La table Fournisseur après modification de l'adresse du fournisseur de NINEA 456321 88
Figure 70 : Syntaxe SQL pour la suppression des données d'une table 89
Figure 71 : Exemple de suppression des données de table Fournisseur 89
Figure 72 : La table Fournisseur après suppression du fournisseur de NINEA 456321 89

15 - Table des matières

A PROPOS I

1 - Organisation pédagogique i
2 - Unité d'enseignement i
3 - Elément constitutif i
4 - Charge de travail i
5 - Modalités ii
6 - Objectifs généraux ii
7 - Mots-clés ii
8 - Prérequis ii
9 - Test d'entrée ii
10 - Références Bibliographiques ii
11 - Déroulement de l'apprentissage iii
12 - Graphe de précédences entre les objectifs iv
13 - Consignes de travail iv
14 - Liste des figures iv
15 - Table des matières vi

SEQUENCE 1 : DES DONNEES A UNE BASE DE DONNEES 1

OBJECTIF 1.1 : DECRIRE LA NOTION DE DONNEE 2

1 - Définition 2
2 - Types de données 2
Exercice 01 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{11}(100%) 3

OBJECTIF 1.2 : MONTRER L'INTERET D'UN FICHIER 4

1 - Définition 4
2 - Types de fichiers 4
Exercice 02 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{12}(100%) 6

OBJECTIF 1.3 : SPECIFIER L'INTERET DE PASSER D'UN FICHIER A UNE BASE DE DONNEES 7

1 - Fichier issu d'un logiciel de traitement de texte 7
2 - Fichier issu d'un logiciel tableur 7
Exercice 03 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{13}(100%) 8

SEQUENCE 2 : GENERALITES SUR LES BASES DE DONNEES 9

OBJECTIF 2.1 : EXPLIQUER UNE BASE DE DONNEES 10

1 - Définition 10
2 - Objectifs 11
3 - Typologie 12
4 - Notion de schéma de données 15
5 - Notion d'instance de données 15
Exercice 04 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{21}(100%) 15

OBJECTIF 2.2 : DECRIRE LES ETAPES DE LA CONCEPTION D'UNE BASE DE DONNEES 16

1 - Analyse de l'existant et des besoins 16

2 - Modélisation conceptuelle des données 16

3 - Modélisation logique des données 17

4 - Implémentation de la base de données 18

5 - Exemple d'application : e-Commerce 18

Exercice 05 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{22}(100%) 19

OBJECTIF 2.3 : DECRIRE UN SYSTEME DE GESTION DE BASE DE DONNEES (SGBD) 20

1 - Définition 20

2 - Objectifs 20

3 - Architecture fonctionnelle 22

4 - Norme ANSI/SPARC 24

5 - Quelques exemples 25

Exercice 06 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{23}(100%) 26

SEQUENCE 3 : MODELISATION CONCEPTUELLE DES DONNEES : MODELE ENTITE-ASSOCIATION 27

OBJECTIF 3.1 : RECENSER LES DONNEES ELEMENTAIRES DU MONDE A MODELISER POUR CONSTITUER UN DICTIONNAIRE DE DONNEES 28

1 - Définition du dictionnaire de données 28

2 - Epuration du dictionnaire de données 29

3 - Formalisme du dictionnaire de données 30

4 - Exemple de dictionnaire de données du système d'information de gestion d'inscription des étudiants dans une université 30

5 - Exemple d'application : e-Commerce 31

Exercice 07 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{31}(20%) 32

Exercice 08 : Seuil de validation : 60% Titre : Dictionnaire de données de la gestion d'un institut de formation Objectifs visés : O_{31}(80%) 32

Exercice 09 : Seuil de validation : 60% Titre : Dictionnaire de données de la gestion des états civils Objectifs visés : O_{31}(80%) 32

Exercice 10 : Seuil de validation : 60% Titre : Dictionnaire de données de la gestion d'une chaîne hôtelière Objectifs visés : O_{31}(80%) 32

OBJECTIF 3.2 : CONSTITUER LES ENTITES A PARTIR DES DONNEES ELEMENTAIRES 33

1 - Entité 33

2 - Dépendance fonctionnelle 36

3 - Identifiant d'entité 38

4 - Exemple d'application : e-Commerce 40

Exercice 11 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{32}(20%) 40

Exercice 12 : Seuil de validation : 60% Titre : Entités de la gestion d'un institut de formation Objectifs visés : O_{31}(80%) 40

Exercice 13 : Seuil de validation : 60% Titre : Entités de la gestion des états civils Objectifs visés : O_{31}(80%) 40

Exercice 14 : Seuil de validation : 60% Titre : Entités de la gestion d'une chaîne hôtelière Objectifs visés : O_{31}(80%) 40

OBJECTIF 3.3 : METTRE LES ENTITES EN RELATION POUR CONSTITUER LE MODELE ENTITE-ASSOCIATION 41

1 - Association 41

2 - Détermination des associations 42

3 - Cardinalité d'une association 44

4 - Typologie des associations 47

5 - Contraintes d'intégrité 50

6 - Quelques logiciels pour concevoir le modèle entité-association 51

7 - Exemple d'application : e-Commerce 51

Exercice 15 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{33}(20%) 51

Exercice 16 : Seuil de validation : 60% Titre : Modèle Entité-Association de la gestion d'un institut de formation Objectifs visés : O_{31}(80%) 52

Exercice 17 : Seuil de validation : 60% Titre : Modèle Entité-Association de la gestion des états civils Objectifs visés : O_{31}(80%) 52

Exercice 18 : Seuil de validation : 60% Titre : Modèle Entité-Association de la gestion d'une chaîne hôtelière Objectifs visés : O_{31}(80%) 52

SEQUENCE 4 : MODELISATION LOGIQUE DES DONNES : MODELE RELATIONNEL 53

OBJECTIF 4.1 : DECRIRE LES PRINCIPES DU MODELE RELATIONNEL 54

1 - Définition 54

2 - Quelques opérations sur les ensembles 55

3 - Objectifs 55

Exercice 19 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{41}(100%) 56

OBJECTIF 4.2 : DECRIRE LA NOTION DE RELATION 57

1 - Définition 57

2 - Eléments du modèle relationnel 57

3 - Quelques opérations sur les relations 58

Exercice 20 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{42}(100%) 59

OBJECTIF 4.3 : PASSER D'UN MODELE ENTITE-ASSOCIATION A UN MODELE LOGIQUE DE DONNEES RELATIONNEL 60

1 - Règles sur les appellations 60

2 - Règles de passage des entités à relations 60

3 - Règles de passage des associations à relations 60

4 - Contraintes d'intégrité 65

5 - Exemple d'application : e-Commerce 66

Exercice 21 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{43}(20%) 66

Exercice 22 : Seuil de validation : 60% Titre : Modèle Logique de Données Relationnel de la gestion d'un institut de formation Objectifs visés : O_{31}(80%) *66*

Exercice 23 : Seuil de validation : 60% Titre : Modèle Logique de Données Relationnel de la gestion des états civils Objectifs visés : O_{31}(80%) *66*

Exercice 24 : Seuil de validation : 60% Titre : Modèle Logique de Données Relationnel de la gestion d'une chaîne hôtelière Objectifs visés : O_{31}(80%) *66*

OBJECTIF 4.4 : NORMALISER UN SCHEMA RELATIONNEL 67

1 - Introduction *67*

2 - Première Forme Normale :1FN *67*

3 - Deuxième Forme Normale :2FN *68*

4 - Troisième Forme Normale :3FN *69*

5 - Forme Normale de Boyce-Codd : FNBC ou 3,5FN *70*

Exercice 25 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{44}(20%) *71*

SEQUENCE 5 : INTERROGATION DES BASES DE DONNEES : L'ALGEBRE RELATIONNELLE 72

OBJECTIF 5.1 : ECRIRE DES REQUETES D'INTERROGATION DE LA BASE DE DONNEES EN ALGEBRE RELATIONNELLE 73

1 - Les opérateurs unaires *73*

2 - Les opérateurs binaires *75*

3 - Les opérateurs composés : les jointures *78*

Exercice 26 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{51}(20%) *79*

Exercice 27 : Seuil de validation : 60% Titre : Requêtes d'interrogation de la gestion d'un institut de formation Objectifs visés : O_{51}(80%) *79*

Exercice 28 : Seuil de validation : 60% Titre : Requêtes d'interrogation de la gestion des états civils Objectifs visés : O_{51}(80%) *79*

Exercice 29 : Seuil de validation : 60% Titre : Requêtes d'interrogation de la gestion d'une chaîne hôtelière Objectifs visés : O_{51}(80%) *80*

SEQUENCE 6 : IMPLEMENTATION D'UNE BASE DE DONNEES : MODELE PHYSIQUE 81

OBJECTIF 6.1 : DECRIRE LES PRINCIPES DE LA GESTION INFORMATISEE D'UNE BASE DE DONNEES 82

1 - Le langage SQL (Structured Query Language) *82*

2 - Les différentes étapes d'implémentation d'une base de données *84*

3 - Exploitation d'une base de données *85*

Exercice 30 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{61}(100%) *85*

OBJECTIF 6.2 : IMPLEMENTER UNE BASE DE DONNEES 86

1 - Téléchargement et installation du SGBD *86*

2 - Création de la base de données *86*

3 - Exemple d'application : e-Commerce *87*

4 - Insertion des données dans la base de données *87*

5 - Modification des données dans la base de données *88*
6 - Suppression des données dans la base de données *89*
Exercice 31 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{62}(25%) *89*
Exercice 32 : Seuil de validation : 60% Titre : Modèle Physique de Données de la gestion d'un institut de formation Objectifs visés : O_{52}(20%) *89*
Exercice 33 : Seuil de validation : 60% Titre : Modèle Physique de Données de la gestion des états civils Objectifs visés : O_{62}(20%) *89*
Exercice 34 : Seuil de validation : 60% Titre : Modèle Physique de Données de la gestion d'une chaîne hôtelière Objectifs visés : O_{62}(20%) *89*
Exercice 35 : Seuil de validation : 60% Titre : Manipulation des données de la gestion d'un institut de formation Objectifs visés : O_{62}(60%) *89*
Exercice 36 : Seuil de validation : 60% Titre : Manipulation des données de la gestion des états civils Objectifs visés : O_{62}(60%) *89*
Exercice 37 : Seuil de validation : 60% Titre : Manipulation des données de la gestion d'une chaîne hôtelière Objectifs visés : O_{62}(60%) *89*
OBJECTIF 6.3 : INTERROGER UNE BASE DE DONNEES EN LANGAGE SQL 90
1 - La sélection ou restriction *91*
2 - La projection *91*
3 - Le renommage *91*
4 - L'union *92*
5 - L'intersection *92*
6 - La différence *92*
7 - Le produit cartésien *93*
8 - La jointure *93*
Exercice 38 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{63}(20%) *96*
Exercice 39 : Seuil de validation : 60% Titre : Interrogation des données de la gestion d'un institut de formation Objectifs visés : O_{63}(60%) *96*
Exercice 40 : Seuil de validation : 60% Titre : Interrogation des données de la gestion des états civils Objectifs visés : O_{63}(60%) *97*
Exercice 41 : Seuil de validation : 60% Titre : Interrogation des données de la gestion d'une chaîne hôtelière Objectifs visés : O_{63}(60%) *97*

Séquence 1 : Des données à une Base de Données

OBJECTIF 1.1 : DÉCRIRE LA NOTION DE DONNÉE

Durée : 04 heures	Seul de validation : 60%

1 - Définition

D'après le dictionnaire Larousse, une donnée est ce (1) qui est connu ou admis comme tel, sur lequel on peut fonder un raisonnement ; (2) qui sert de point de départ pour une recherche. La donnée est un élément fondamental qui sert de matière première dans les échanges entre les humains, matériels (smartphone, ordinateur, …), etc. Quel que soit le domaine d'activités ou la science, la donnée est définie avec des nuances insignifiantes.

En **mathématiques**, une donnée est une quantité connue dans l'énoncé d'un problème et qui sert à trouver la solution.

En **informatique**, une donnée est la représentation d'une information dans un programme : soit dans le texte du programme (code source), soit en mémoire durant l'exécution. Les données, souvent codées, décrivent les éléments du logiciel tels qu'une entité (chose), une interaction, une transaction, un évènement, un sous-système, etc. Il ne faut surtout pas confondre la notion de donnée et celle d'information. Une **information** se diffère d'une donnée par le fait que la donnée est un élément brut, dépourvue de tout raisonnement, supposition, constatation et probabilité alors qu'une information est une donnée interprétée. Pour passer de donnée à information, il faut un contexte de la donnée qui créé une valeur ajoutée.

En **science des données**, une donnée est ce qui est connu et qui sert de point de départ à un raisonnement ayant pour objet la détermination d'une solution à un problème en relation avec cette donnée. Cela peut être une description élémentaire d'une réalité, le résultat d'une comparaison entre deux événements du même ordre (mesure) soit en d'autres termes une observation ou une mesure. De nos jours, on parle de **Big Data** ou **Données massives** pour caractériser de grands volumes de données variées et en constante production.

2 - Types de données

Une donnée appartient toujours à un ensemble de valeurs (domaine de valeurs) que nous appelons **type**. On distingue des types prédéfinis (types simples) et des types composés (types complexes).

2.1 - Types prédéfinis ou types simples

- **Entier** : nombre sans partie décimale pouvant être un entier naturel (ℕ) ou un entier relatif (ℤ).
- **Réel** : nombre avec partie décimale pouvant être un nombre décimal (𝔻), un nombre rationnel (ℚ) ou un nombre réel (ℝ).
- **Caractère** : la valeur est un caractère numérique (0, 1, … 9), ou un caractère alphabétique (a, b, … z), ou un caractère spécial (#, +, /, \, *, $, €, =, %, …) entre côtes.
- **Chaîne de caractères** : suite de caractères entre côtes pouvant être vide.
- **Booléen** : la valeur dans un ensemble à seulement 2 valeurs soit {Vrai, Faux} ou {Oui, Non} ou autres {0, 1}.

2.2 - Types composés ou types complexes

Ce sont des types composés à partir des types prédéfinis comme par exemple le nombre complexe, l'mage, le son, la vidéo, etc.

Exercice 01 : **Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{11}(100%)**

OBJECTIF 1.2 : MONTRER L'INTÉRÊT D'UN FICHIER

Durée : 04 heures	Seul de validation : 60%

1 - Définition

Les données sont en général créées çà et là en fonction des évènements, interactions, activités, … Par contre elles doivent être regroupées en des fiches en fonction de leurs liens sémantiques c'est-à-dire que le regroupement des données se fera selon un but bien précis pour avoir un ensemble cohérent. C'est ainsi que l'on peut avoir la fiche du personnel, la fiche d'une vente, la fiche d'un patient dans une structure sanitaire, la fiche de notes des étudiants, la fiche d'une leçon, la fiche d'une recette de cuisine, …

En informatique, cette fiche est appelée **fichier** et est définie comme un ensemble structuré ou organisé d'informations désigné par un nom précis et enregistré dans une unité de stockage comme un objet et exploité par des programmes. Sans le fichier, on aurait les données et/ou informations pêle-mêle sans organisation ou structuration rendant difficile leur mise en relation et exploitation dans des buts précis. En revanche, une donnée peut se retrouver dans plusieurs fichiers avec des sémantiques pouvant être différentes. Par exemple, votre filiation (prénom, nom, date de naissance, …) peut appartenir au fichier de boursier et aussi au fichier de notes en base de données. Votre filiation est utilisée dans les 2 fichiers avec des sens différents : dans le premier cas, cela veut dire que vous êtes boursier et dans le second cas, cela veut dire que vous avez été évalué en base de données.

En langage naturel, les termes fichier et dossier sont utilisés souvent sans différenciation mais en informatique, un **dossier** ou **répertoire** est regroupement de fichiers selon un ou des critères.

2 - Types de fichiers

Les fichiers informatiques sont créés à partir des logiciels (programmes) et il existe plusieurs types de fichiers.

2.1 - Fichier exécutable

Fichier qui contient des instructions ou commandes exécutables par l'ordinateur. Un fichier exécutable est un programme informatique et est caractérisé par l'une des extensions suivantes :

- **.exe** : Extension d'une application (programme ou logiciel) destinée à Microsoft Wndows
- **.sh** : Extension des fichiers de scripts sous Linux
- **.bat** : Extension des fichiers de commandes sous Microsoft Windows
- **…**

2.2 - Fichier image

Fichier contenant des images, dessins, pictogrammes, photos, graphismes, … et caractérisé par l'une des extensions suivantes :

- .**gif** : Graphics Interchange Format
- **.bmp** : BitMaP
- **.jpeg** ou **.jpg** : Joint Photographic Experts Group
- **.tiff** : Tag(ged) Image File Format
- **.png** : Portable Network Graphics
- **…**

2.3 - Fichier audio

Fichier contenant du son codé sous une forme exploitable par l'ordinateur et caractérisé par l'une des extensions suivantes :

- **.mp3**
- **.wav**
- **.au**
- **.ram**
- **.oga**

2.4 - Fichier vidéo

Fichier contenant de la vidéo (combinaison du son, image, animation, texte) codée sous une forme exploitable par l'ordinateur et caractérisé par l'une des extensions suivantes :

- **.mp4**
- **.avi**
- **.mpeg**
- **.mov**

2.5 - Fichier document

- **.doc, .docx** : Extensions des fichiers produits avec le logiciel Microsoft Word (Traitement de texte)
- **.ppt, .pptx** : Extensions des fichiers produits avec le logiciel Microsoft PowerPoint (Présentation Assistée par Ordinateur)
- **.xls, .xlsx** : Extensions des fichiers produits avec le logiciel Microsoft Excel (Traitement de données)
- **.pub** : Extension des fichiers produits avec le logiciel Microsoft Publisher (Publication Assistée par Ordinateur)
- **.pdf** : Extension des Fichiers Portables c'est-à-dire lisible sur n'importe quel ordinateur
- **.html** : Extension des fichiers web ouvrables avec un navigateur web (Google Chrome, Microsoft Explorer, Mozilla Firefox, …)
- **…**

2.6 - Fichier texte

Fichier contenant du texte brut sans indication typographique (pas de mise en forme ni mise en page). Généralement utilisé dans les contextes suivants :

- Echange d'informations entre 2 programmes différents : **.txt**, **.csv**, **…**
- Données de configuration d'un logiciel : **.ini**, **.conf**, **…**
- Code source d'un programme : **.py**, **.c**, **.pas**, **.java**, **.html**, **.xml**, **.php**, **.js**, **…**
- Journalisation (historique des évènements) d'un logiciel : **.log**, **…**

2.7 - Fichier compressé

Fichier recodé en vue de rendre le plus petit en taille sans perte d'information que le fichier original. Le programme chargé d'assurer cette tâche s'appelle un **compresseur**.

- **.zip**
- **.gz**
- **.rar**
- **.tar**

Exercice 02 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{12}(100%)

OBJECTIF 1.3 : SPÉCIFIER L'INTÉRÊT DE PASSER D'UN FICHIER À UNE BASE DE DONNÉES

Durée : 04 heures	Seul de validation : 75%

Un fichier contient un ensemble de données structurées mais il n'existe pas de possibilités d'automatiser leur mise à jour, suppression, interrogation et ajout de nouvelles données. Citons quelques exemples de types de fichiers et leurs limites.

1 - Fichier issu d'un logiciel de traitement de texte

Microsoft Word, Open Office Writer, LibreOffice Writer, Google Docs… sont des exemples de logiciel de traitement de texte.

Un fichier issu d'un logiciel de traitement contient un ensemble de données et/ou d'informations formatées (mise en forme, mise en page, …) et structurées généralement en titres (partie, chapitre, titre, sous-titre, …). Avec les logiciels de traitement de texte, les opérations sont réalisées ainsi :

- La suppression automatique d'une donnée est impossible ; il faut la chercher, la sélectionner et la supprimer manuellement.
- La modification de toutes les occurrences d'une donnée est automatisée par la fonction *« Rechercher et Remplacer »*.
- L'ajout d'une nouvelle donnée se fait manuellement car il faut chercher sa position dans le texte et la saisir à l'emplacement convenable.
- L'interrogation se fait par la fonction « *Rechercher* » mais ne donne pas en une liste des occurrences de la donnée cherchée ; il faut utiliser les boutons *Suivant* et *Précedent* pour parcourir les différentes occurrences trouvées.

2 - Fichier issu d'un logiciel tableur

Microsoft Excel, Open Office Calc, LibreOffice Calc, Google Sheets, … sont des exemples de logiciel tableur.

Un fichier issu d'un logiciel tableur est un classeur contenant des feuilles où chaque feuille est un tableau de plusieurs colonnes et lignes. L'information ou la donnée est soit une formule, soit une donnée brute et est contenue dans une cellule ou case (intersection entre une colonne et une ligne). Avec les logiciels de tableur, les opérations sont réalisées ainsi :

- La suppression automatique d'une donnée est impossible ; il faut la chercher, la sélectionner et la supprimer manuellement.

- La modification de toutes les occurrences d'une donnée est automatisée par la fonction *« Rechercher et Remplacer »*.
- L'ajout d'une nouvelle donnée se fait manuellement car il faut chercher sa position dans le classeur et la saisir à l'emplacement convenable.
- L'interrogation se fait par la fonction « *Rechercher* » mais ne donne pas en une liste des occurrences de la donnée cherchée ; il faut utiliser les boutons *Suivant* et *Précédent* pour parcourir les différentes occurrences trouvées.
- Il faut toutefois noter que la plupart de logiciels tableurs permettent l'automatisation de certaines opérations moyennant l'écriture des macros (petits programmes embarqués)

Exercice 03 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{13}(100%)

SÉQUENCE 2 : GÉNÉRALITÉS SUR LES BASES DE DONNÉES

OBJECTIF 2.1 : EXPLIQUER UNE BASE DE DONNÉES

Durée : 12 heures	Seul de validation : 50%

1 - Définition

Une base de données est un ensemble d'informations structurées, semi-structurées ou non structurées d'un système d'information organisé en vue de faciliter leur exploitation (interrogation et mise à jour) et gérer leur stockage.

Un système d'information est l'ensemble des moyens (organisation, acteurs, procédures, systèmes informatiques) nécessaires à la saisie, traitement, mémorisation, classement, recherche, et diffusion des informations. Il a pour objectif de restituer, aux différents membres de l'entreprise ou organisme, les informations sous une forme directement utilisable au moment opportun, afin de faciliter le bon de fonctionnement opérationnel et la prise de décisions aux différents niveaux.

Pour obtenir une base de données, il faut modéliser une situation réelle et le représenter par un modèle selon un niveau d'abstraction désiré.

1.1 - Donnée structurée

Une donnée structurée est une donnée formatée ou représentée selon une structure de données précise. Vous avez vu dans le cours d'algorithmique et programmation la notion de structure de données. C'est le type de données le plus ancien ayant l'avantage de faciliter le contrôle et l'exploitation des données en se basant sur la structure. Par contre sa structure limite sa flexibilité et son usage à ce qui a été défini.

Comme exemple, on peut citer les fiches d'étudiants, les données d'un point de vente, les fiches de paie des employés, …

1.2 - Donnée non structurée

Une donnée non structurée est une donnée sous sa forme brute absolue sans traitement. Les données non structurées peuvent être tout ce qui n'est pas dans un format ou structure spécifique. Comme avantage, on peut citer la liberté de spécifier le format selon le besoin ce qui permet un grand nombre de cas d'utilisation car la finalité des données est adaptable. Par contre, c'est une donnée difficile à exploiter du fait qu'il faut une expertise en science de données pour leur préparation et leur analyse.

Comme exemple, on peut citer les fichiers journaux des applications, les publications sur les réseaux sociaux et leurs commentaires, les discussions des forums ou chats ou emails, …

1.3 - Donnée semi-structurée

Une donnée semi-structurée est une donnée non structurée qui comporte des métadonnées avec certaines caractéristiques. La métadonnée contient des informations sur les données permettant de mieux les cataloguées, analysées et faire faire des recherches efficacement par rapport aux données non structurées. Les données semi-structurées sont présentées comme des données intermédiaires entre les données structurées et non structurées et bénéficient en partie de leurs avantages.

Comme exemple, on peut citer les fichiers CSV (Comma-Separated Values) contenant des données séparées par un délimitateur, les fichiers de données avec des balises (html, xml, …), …

2 - Objectifs

2.1 - Structuration, sauvegarde et exploitation des données d'un système d'information

La structuration est le fait qu'une base de données sert à représenter les données d'un système d'information et les liens sémantiques entre elles.

La sauvegarde est le fait de garder de façon pérenne les données sur une unité de stockage ou mémoire de masse.

L'exploitation des données est le fait de définir des moyens d'interrogation, d'ajout, de modification et de suppression des données.

2.2 - Cohérence ou intégrité des données

La cohérence des données est la capacité de maintenir les données dans un état correct après une opération d'ajout, modification ou suppression.

Pour garantir la cohérence des données, il faut définit un ensemble de règles que les données doivent respectées en tout temps.

Exemples de règles pour la cohérence des données :

- Dans un système d'information de gestion d'une pharmacie, la quantité en stock ne doit pas être négative. Cela veut dire par exemple que la vente d'un médicament ne sera possible que si la quantité en stock est supérieure ou égale à la quantité à vendre.

- Dans un système d'information de gestion pénitentiaire, un détenu ne doit pas avoir plus de 5 visites par mois.

2.3 - Non redondance des données

La redondance des données est le fait de les répéter inutilement dans la base de données. Les données redondantes entraînent des problèmes de mise à jour des données car lors des modifications d'une donnée, il est possible d'oublier certaines occurrences ce qui rend les données incohérentes ou, il faut chercher et s'assurer de la modification de toutes les occurrences de la donnée, ce qui rend le traitement lent.

Pour éviter la redondance des données, une base de données doit définir une structuration rigoureuse des données afin que chacune ne soit pas répétitive inutilement.

2.4 - Exhaustivité des données

L'exhaustivité des données est le fait que la base de données soit complète c'est-à-dire doit contenir toutes les informations nécessaires afin de répondre aux besoins des utilisateurs.

3 - Typologie

3.1 - Selon l'emplacement

a) Base de données centralisée

C'est une base de données dans laquelle les données sont stockées dans un seul ordinateur ou serveur central. La centralisation des données facilite leur stockage et conservation. En outre, ça peut être un goulot d'étranglement vue de nombreux utilisateurs de la base de données.

b) Base de données distribuée

C'est une base de données dans laquelle les données sont stockées dans plusieurs unités de stockage qui ne se trouvent pas au même emplacement physique. Ces différentes unités de stockage sont indexées et gérées par un SGBD central. L'avantage d'une base de données distribuée est la répartition ou l'équilibrage des charges au niveau de chaque ordinateur hébergeant une partie des données, évitant le goulot d'étranglement. En outre, mettre à jour les données est une tâche complexe et coûteuse en temps. De plus la conception d'une base de données distribuée est plus complexe qu'une base de données centralisée.

3.2 - Selon l'usage

a) Base de données opérationnelle

Une base de données opérationnelle ou **OLTP** (**OnLine Transaction Processing : Traitement transactionnel en ligne**) est une base de données où l'accent est mis sur la capacité de traitement et la vitesse de réponse de plusieurs opérations simultanément pour satisfaire des requêtes en temps réel.

b) Base de données d'analyse

Une base de données d'analyse ou **OLAP** (**OnLine Analytical Processing : Traitement analytique en ligne**) est une base de données où l'accent est mis sur la capacité à stocker et analyser de grandes masses de données complexes issues des bases de données opérationnelles. Cette analyse permet de donner des statistiques, des prévisions, … bref des informations précieuses pour la prise de décision dans le management (catastrophe, météo, système sanitaire, ressource humaine, approvisionnement, …). Ces bases de données sont généralement appelées des entrepôts de données (Data Warehouse).

3.3 - Selon la structuration

Il existe quatre types de base de données qui correspondent à l'évolution des bases de données.

a) Base de données hiérarchique

Les premières bases de données structuraient l'information/donnée de façon hiérarchique où chacune dépendait (lien de parenté) d'une seule information/donnée. La structure de la base se présente comme un arbre. Le fait qu'une donnée ne dépende que d'une seule appelée donnée parent présente des limites car en général, une donnée est en relation avec plusieurs autres et la sémantique de leur relation n'est pas qu'une relation père-fils.

Exemples :

- ADABAS (Adaptable DAta BAse System) de Software AG
- IMS (Information Management System) de IBM (International Business Machines Corporation) qui fut utilisé pour le programme Apollo
- Base de registre de Microsoft

b) Base de données réseau

Pour corriger les limites des bases de données hiérarchiques où chaque donnée n'avait qu'un parent, Charles William Bachman créé les bases de données réseau où chaque donnée

peut être liée à plusieurs autres. Le modèle réseau est une amélioration du modèle hiérarchique permettant de passer d'une structure en arbre à une structure en graphe.

Une très forte dépendance entre les données et les programmes du modèle réseau empêche l'utilisation des bases de données réseau dans plusieurs systèmes informatiques.

c) Base de données relationnelle

C'est le type de base de données le plus utilisé depuis les années 1980 à nos jours. Les bases de données relationnelles reposent sur l'**algèbre relationnelle**[1] qui est une théorie mathématique inventée par Edgar Franck Codd. L'information est représentée dans les tables (tableaux) où chaque colonne représente une caractéristique de la donnée et une ligne représente une occurrence de la donnée (occurrence de toutes ses caractéristiques).

d) Base de données objet

S'appuyant sur le principe de la **Programmation Orientée Objet** où l'objet est l'encapsulation des données et traitements en une seule entité, les bases de données objet sont encore à un stade embryonnaire car leur développement ne suit pas le modèle objet avec les différents concepts d'héritage, encapsulation, polymorphisme, … Beaucoup d'espoirs sont fondés sur ce type de base de données connaissant la plus-value de la Programmation Orientée Objet.

e) Autres types

- **Base de données non relationnelle ou NoSQL (Not Only SQL)**

Le langage SQL (Structured Query Language) domine largement le mode des bases de données avec les bases de données relationnées. Mais, un courant de pensée propose le concept de « **Not Only SQL : Pas seulement SQL** » pour la gestion des bases de données. L'idée est de dénormaliser une base de données relationnelle pour stocker les données sous forme peu structurée sans suivre un schéma fixe ; des opérateurs comme la jointure entre les tables ne sont plus nécessaires. Le **Big Data**[2] est un exemple typique des bases de données NoSQL.

- **Base de données orientée texte ou base de données dans un fichier plat**
- **Base de données orientée document**

---

[1] https://fr.wikipedia.org/wiki/Alg%C3%A8bre_relationnelle

[2] https://www.piloter.org/business-intelligence/big-data-definition.htm

4 - Notion de schéma de données

Le schéma d'une base de données est la description de la structure (sans les données) des objets (table, vue, champ, clé, …) et règles (contraintes d'intégrité, …) dans un langage formel (SQL en général) pris en charge par le Système de Gestion de Base de Données.

Une fois que le schéma est créé, il est presque statique car il change rarement durant son cycle de vie.

Il existe deux principaux types de schémas de base de données :

- Un **schéma de base de données logique** transmet les contraintes logiques applicables aux données stockées. Il peut définir des contraintes d'intégrité, des vues et des tables.
- Un **schéma de base de données physique** expose la manière dont les données sont stockées physiquement dans un système de stockage en termes de fichiers et d'index.

Le processus de création d'un schéma de base de données est la modélisation des données que nous décrirons plus loin à l'Objectif 2.2 : Décrire les étapes de la conception d'une base de données.

5 - Notion d'instance de données

L'instance d'une base de données est l'ensemble de données de la base à un instant donné. Ainsi, l'instance d'une base de données change au fil du temps puisque les utilisateurs ajoutent, modifient ou suppriment les données constamment.

L'instance d'une base de données est constituée de l'instance de chaque objet (table) de la base de données. Chaque donnée de l'instance doit respecter les structures des données définies dans le schéma de base de données.

Exercice 04 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{21}(100%)

OBJECTIF 2.2 : DÉCRIRE LES ÉTAPES DE LA CONCEPTION D'UNE BASE DE DONNÉES

Durée : 12 heures	Seul de validation : 80%

La conception d'une base de données est un processus qui obéit à une méthodologie rigoureuse en étapes.

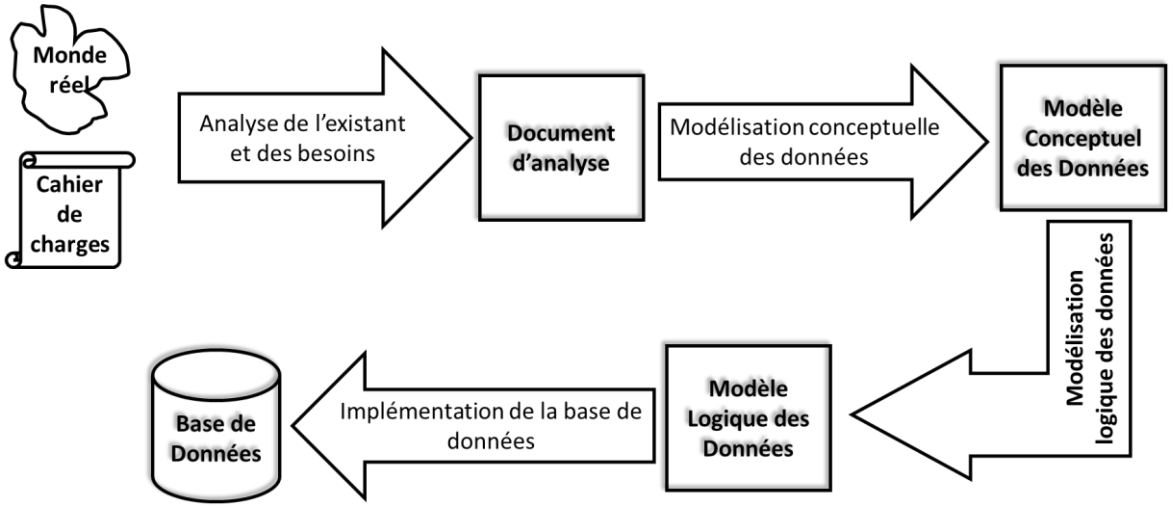


Figure 1 : Processus de conception d'une base de données

1 - Analyse de l'existant et des besoins

Les besoins et la description de l'existant sont contenus dans un cahier de charges qui est un élément fondamental du contrat qui lie le concepteur de base de données au client. Ce document doit être assez précis pour permettre au concepteur de comprendre le monde à modéliser et surtout sa délimitation. L'analyse de l'existant et des besoins est une étape cruciale et délicate. Contrairement aux autres étapes où il existe des formalismes, cette étape ne repose que sur l'expertise et l'ingéniosité du concepteur. Durant cette étape, il faut beaucoup communiquer avec le client en faisant des interviews autant que nécessaire, demandant des documents de référence aux problèmes à résoudre, …

A l'issue de cette étape, on aura un document contenant les besoins, les choix (faits et à faire), les contraintes, la circonscription de la future base de données, …

2 - Modélisation conceptuelle des données

A partir du document d'analyse, il est question de conceptualiser une solution à partir des spécifications formelles non ambiguës de modélisation dans un langage "naturel". Pour cela on peut utiliser soit :

- **Modèle Entité-Association** ou **Diagramme Entité-Association** qui est un langage graphique destiné à la construction du modèle conceptuel d'une base de données, indépendamment du SGBD qui sera utilisé pour gérer celle-ci. Il est à la base de nombreuses méthodes de conception de base de données dont la méthode MERISE.
- **MERISE (Méthode d'Étude et de Réalisation Informatique par les Sous-Ensembles ou pour les Systèmes d'Entreprise)** qui est une méthode d'analyse et de conception des systèmes d'information basée sur le principe de la séparation des données et des traitements. Pour la partie conceptuelle des données, il faut utiliser le Modèle Conceptuel de Données (MCD).
- **UML (Unified Modeling Language : Langage de Modélisation Unifié)** qui est un langage de modélisation graphique à base de pictogrammes conçu comme une méthode normalisée de visualisation dans les domaines du développement logiciel et en conception orientée objet. Pour la partie conceptuelle des données, il faut utiliser le diagramme de classes.

A l'issue de cette étape, on aura le diagramme entité-association qui est une représentation simplifiée du monde réel modélisé.

Dans le cadre de ce cours, nous utiliserons le modèle entité-association inventé par **Peter Chen** dans les années 70. C'est une représentation graphique des données modélisées facilite l'appréhension et la compréhension du système à modéliser. Il est décrit des concepts d'entité, association, propriété, identifiant, cardinalité, rôle, ….

3 - Modélisation logique des données

Le diagramme entité-association issu de l'étape précédente est dans un langage naturel or la finalité est de l'implémenter dans un ordinateur. Il est donc question de passer à une représentation moins naturelle mais propice pour un ordinateur. Cette étape consiste à traduire le modèle conceptuel de données en modèle logique de données en respectant les règles de passage ou de conversion. Les règles à appliquer doivent tenir compte du type de base de données ciblé sans se soucier de son implémentation. C'est une étape mécanique, peu technique que les précédentes ; et à ce titre, il existe plusieurs logiciels capables de le faire.

A cette étape, si le type de base de données choisi est relationnel, il faudra normaliser le modèle obtenu pour minimiser les redondances des données qui entraînent des anomalies sémantiques qui, à leur tour, rendent le traitement automatique des données et la maintenance des bases de données difficiles.

4 - Implémentation de la base de données

L'implémentation consiste à faire des choix techniques en termes de SGBD (Système de Gestion de Base de Données) et décrire dans le langage du SGBD le modèle logique de données obtenu précédemment. Cette étape nécessite des compétences technico-informatiques pour l'implémentation et l'optimisation de la base de données.

5 - Exemple d'application : e-Commerce

Cet exemple sera le fil conducteur durant ce cours. Dans cette section, il est question d'analyser les besoins d'une gestion de site de commerce en ligne et de délimiter le système.

Document d'analyse de l'existant et des besoins

Il vous est demandé de concevoir une base de données pour un site de commerce en ligne. Chaque client doit se créer un compte d'utilisation du site composé d'un login (nom d'utilisateur) et d'un password (mot de passe). Pour distinguer les utilisateurs, le login doit être unique. Pour des raisons de sécurité, le mot de passe est obligatoire et doit avoir une longueur d'au moins 8 caractères. Le client est caractérisé par son nom, prénom, référence, date de naissance, sexe, téléphone et adresse électronique.

Lorsqu'un client arrive sur le site, il créé une commande (panier) devant contenir la quantité de chaque produit à commander. Une commande est caractérisée par un numéro et une date. Les produits sont classés par catégorie et un produit appartient à une seule catégorie. Un produit est caractérisé par son code, libellé (nom) et quantité en stock. Certaines catégories sont des sous-catégories.

Sur le site d'e-commerce, plusieurs clients achètent les mêmes produits presqu'au même moment et de plus un client peut stocker les produits dans son panier et l'acheter plus tard. De ce fait, il peut arriver qu'au moment de la facturation, la quantité commandée ne soit plus suffisante en stock. Le client a le choix de soit enlever ce produit de sa facturation, soit prendre la quantité qui existe. Il pourra sans besoin d'une nouvelle commande se faire facturer les produits non livrés. Dans la facture du client, la quantité livrée doit être précisée. Chaque facture est caractérisée par un numéro, date et appartient à un mode de règlement.

Pour satisfaire sa clientèle, le gestionnaire du site s'approvisionne périodiquement auprès de plusieurs fournisseurs. Pour automatiser le déclenchement d'approvisionnement, chaque produit est caractérisé par un seuil d'alerte. Un approvisionnement est fait chez un seul fournisseur et contient pour chaque produit, une quantité commandée. Chez les fournisseurs,

toutes les quantités commandées sont livrées. Un approvisionnement est caractérisé par son identifiant et date. Un fournisseur est caractérisé par son NINEA (Numéro d'Identification National des Entreprises et des Associations), désignation, adresse, téléphone, adresse électronique et site web.

Exercice 05 : **Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{22}(100%)**

OBJECTIF 2.3 : DÉCRIRE UN SYSTÈME DE GESTION DE BASE DE DONNÉES (SGBD)

Durée : 24 heures	Seul de validation : 50%

1 - Définition

Un Système de Gestion de Base de Données est un logiciel permettant de gérer (créer, modifier, supprimer, sécuriser, exploiter, …) une base de données. A cet effet, il comporte de nombreux modules et interfaces utilisateurs. L'exploitation d'une base de données consiste à réaliser les opérations suivantes sur les données : insertion, interrogation, modification et suppression. La sécurisation de la base de données consiste à créer des utilisateurs et leur octroyer des privilèges (droits) sur les données (tables) et les opérations. Par exemple, un utilisateur peut avoir le droit d'interroger une table mais ne pas avoir le droit de modifier ses données.

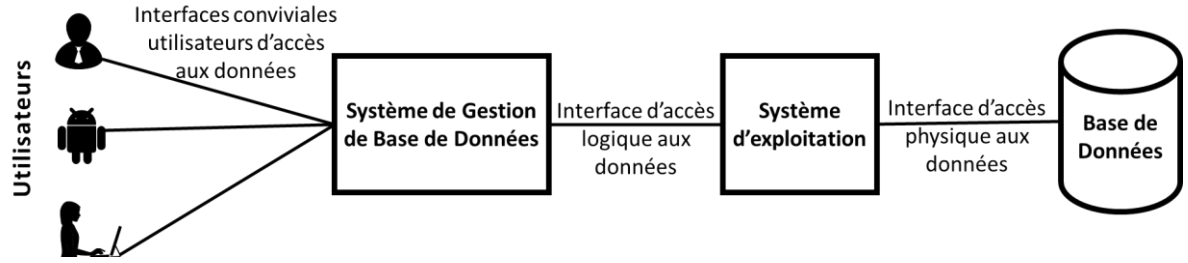


Figure 2 : Position du SGBD par rapport à la base de données

2 - Objectifs

2.1 - Indépendance physique des données

L'indépendance physique permet de modifier le schéma interne (niveau physique) sans modifier le schéma conceptuel (niveau conceptuel). Ainsi la définition des structures des données (niveau conceptuel) est indépendamment des structures de stockage (niveau physique ou interne).

La structure de stockage des données appartient au monde des informaticiens et n'a donc aucun un sens que dans l'univers du système informatique. Le schéma interne décrit un assemblage physique des données en articles, fichiers, chemins d'accès (organisation et méthode d'accès aux fichiers, modes de placement des articles dans les fichiers, critères de tri, chaînages…) sur des mémoires de stockage. Cet assemblage propre au monde informatique doit être basé sur des considérations de performance et de souplesse d'accès. On pourra par exemple ajouter un index, regrouper deux fichiers en un, changer l'ordre ou le codage des

données dans un article, sans mettre en cause les entités et associations définies au niveau conceptuel.

2.2 - Indépendance logique des données

L'indépendance logique permet de modifier les schémas conceptuels (niveau interne ou physique) sans changer les programmes d'application (niveau externe). Ainsi l'ajout ou le retrait de nouveaux éléments n'a aucune influence sur les programmes qui n'y font pas explicitement référence.

2.3 - Indépendance entre données et traitements

L'indépendance entre les données et les traitements vise à permettre une évolution des structures de données sans répercussion sur les programmes d'application utilisant ces données. Pour atteindre cet objectif, il a été mis en place les concepts de schéma et de sous-schéma de données. Le sous-schéma correspond à une vue de la base de données extraite du schéma et intéressant un utilisateur ou un groupe d'utilisateurs. C'est l'interface entre le schéma et le sous-schéma qui garantit cette indépendance.

2.4 - Sécurité des données

Les données ne doivent être accessibles en partie ou en totalité qu'aux utilisateurs ayant des droits. L'administrateur dispose ainsi des outils pour autoriser, contrôler ou enlever les droits d'accès à n'importe qui.

2.5 - Efficacité des accès aux données

Le SGBD doit pouvoir fournir des réponses aux requêtes le plus rapidement possible ; il faut des algorithmes très rapides et des stratégies de gestion des files d'attente. Par exemple, il ne faut pas que la requête courte d'un utilisateur attende la fin d'une longue requête d'un autre utilisateur.

2.6 - Accès concurrent aux données

Le SGBD doit permettre l'accès aux données par plusieurs utilisateurs à la fois tout en s'assurant de la cohérence des données et la gestion des conflits en cas de mises à jour simultanées de la même donnée. Pour cela, il y a des mécanismes de verrouillage permettant de détecter et gérer les conflits.

2.7 - Administration facilitée des données

Un SGBD doit fournir des outils pour décrire les données, à la fois leurs structures de stockage et leurs présentations externes. Il doit permettre le suivi de l'adéquation de ces structures aux besoins des applications et autoriser leur évolution aisée. Ces tâches sont réservées à l'administrateur de la base de données.

2.8 - Résistance aux pannes

Les systèmes informations sont sujets aux pannes (électrique, matérielle, logicielle, réseau informatique, …) durant leur utilisation. Par exemple, vous voulez approvisionner votre stock dans le cas d'une boutique. D'après la structure de la base de données, il faut saisir l'approvisionnement (fournisseur, quantité, date, observation, …) de chaque produit et mettre à jour leur quantité en stock totale. Imaginez-vous qu'après la saisie des approvisionnements, qu'il y ait une panne ! Les données se trouveront dans un état incohérant car la quantité en stock totale ne reflète pas la réalité.

Le SGBD doit pouvoir gérer de telles situations par une transaction qui doit être soit validée si toutes les opérations ont été réussies ; soit détruite au cas où l'une des requêtes échoue. La destruction de la transaction annule les actions des précédentes requêtes qu'elle contient.

2.9 - Manipulation des données

Les utilisateurs doivent être capables d'écrire leurs requêtes à partir des langages évolués (langages proches du langage naturel) sans faire référence aux éléments techniques de la base de données.

3 - Architecture fonctionnelle

L'architecture fonctionnelle de référence de la plupart des SGBD contient 5 composants prenant en charge une requête de l'utilisateur jusqu'à son exécution.

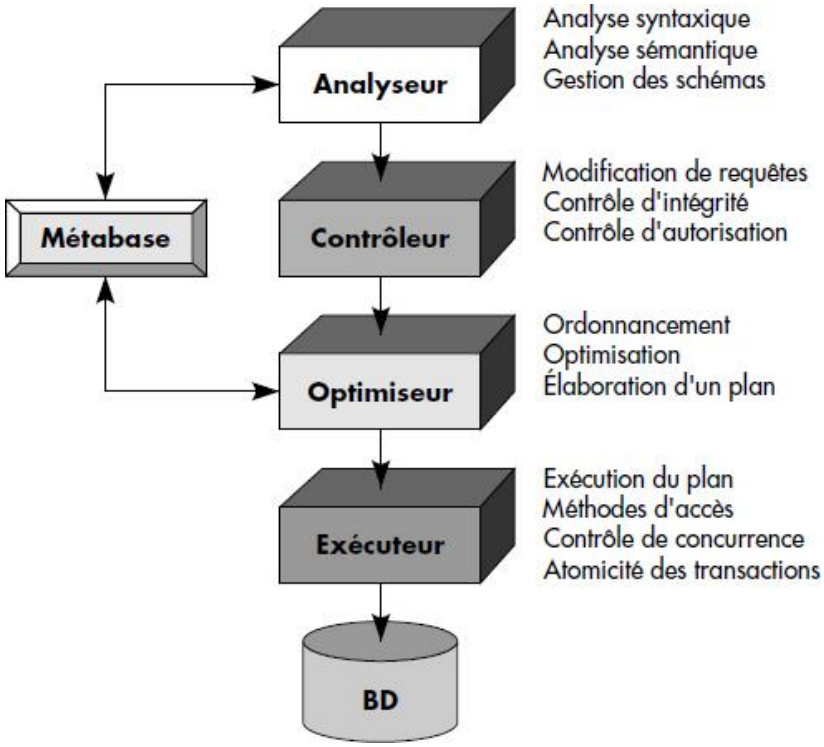


Figure 3 : Architecture d'un Système de Gestion de Base de Données

3.1 - Métabase

La métabase est un dictionnaire de données permettant de décrire les autres bases de données. Elle contient des commandes de définition du schéma des tables et la définition des vues. Lorsqu'une requête est écrite, c'est cette métabase qui permet de la valider en termes de respect de la structure des données (syntaxique et sémantique).

3.2 - L'analyseur de requêtes

L'analyseur se base sur la métabase pour analyser les requêtes et s'assurer qu'elles sont valides syntaxiquement (respect de l'orthographe et de la grammaire) et sémantiquement (conformité à la vue référencée ou au schéma). Si la requête est correcte alors elle est traduite du langage évolué à un langage informatique où entre autres les noms des objets sont remplacés par des références informatiques (adresses mémoires).

3.3 - Le contrôleur de requêtes

A partir d'une requête correcte venant de l'analyseur, le contrôleur s'assure les fonctions suivantes :

- Contrôle des droits de lecture et/ou d'écriture sur les tables de la requête par l'utilisateur
- Contrôle d'intégrité des données lors des mises à jour

- Au cas où la requête fait référence à une vue, cette vue doit être remplacée par la requête ayant permise sa création. Ainsi, la requête ne contiendra que les objets du schéma de la base de données

3.4 - L'optimiseur de requêtes

L'optimiseur élabore un plan d'accès optimisé pour traiter la requête. Pour se faire, il décompose en général la requête en opérations d'accès et choisit un ordre d'exécution optimal. Il choisit aussi les méthodes d'accès à utiliser. Pour effectuer les meilleurs choix, l'optimiseur s'appuie souvent sur un modèle de coût qui permet d'évaluer le coût d'un plan d'accès avant son exécution. Le résultat de l'optimisation (le plan d'accès optimisé) peut être sauvegardé en mémoire pour des exécutions multiples ultérieures ou exécuté directement puis détruit.

3.5 - L'exécuteur de plans

L'exécuteur exécute le plan d'accès élaboré par l'optimiseur. Pour cela, il s'appuie sur les méthodes d'accès qui permettent d'accéder aux fichiers via des index et/ou des liens. C'est aussi à ce niveau que sont gérés les problèmes de concurrence d'accès et d'atomicité de transactions. Les techniques utilisées dépendent beaucoup de l'architecture opérationnelle du SGBD qui s'exprime en termes de processus et de tâches.

4 - Norme ANSI/SPARC

La norme ANSI/SPARC est une architecture à 3 niveaux qui a été principalement définie pour permettre l'accès aux données indépendamment de leur représentation physique.

Ses objectifs sont :

- **Personnalisation des vues indépendantes** : chaque utilisateur doit avoir accès aux mêmes données (sous réserve de droits), mais avec une vue personnalisée différente des données.
- **Transparence des détails de stockage physiques aux utilisateurs** : les utilisateurs ne doivent pas avoir à gérer les détails de stockage de base de données.
- **Souplesse des vues par rapport au schéma de la base de données** : l'administrateur de la base de données doit être en mesure de modifier les structures de stockage de la base de données sans affecter les vues des utilisateurs.

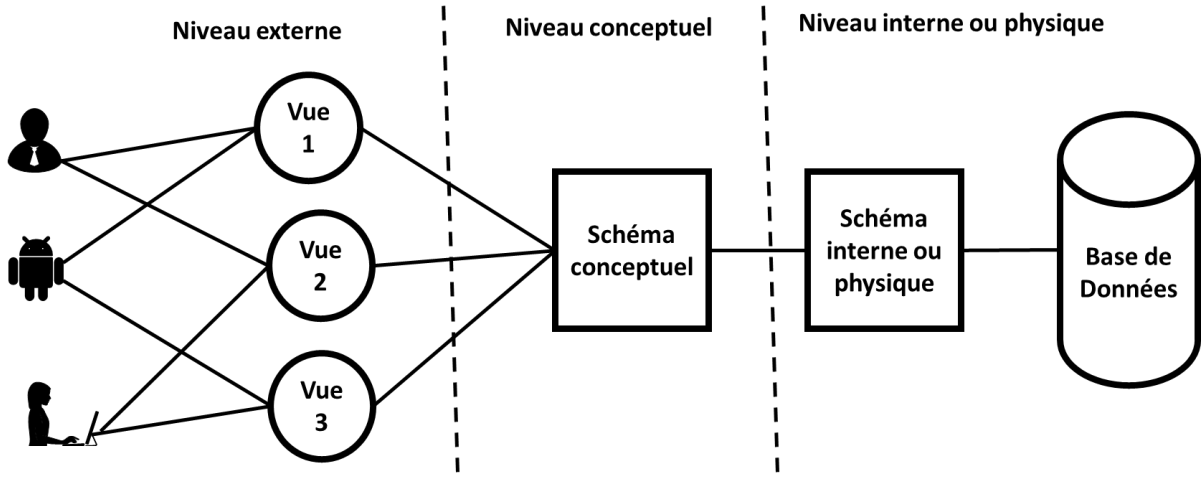


Figure 4 : Architecture à 3 niveaux d'ANSI/SPARC

4.1 - Niveau externe

Il est composé des vues utilisateurs qui sont des parties de la base de données pertinentes pour un utilisateur ou un groupe d'utilisateurs.

4.2 - Niveau conceptuel

Il décrit la structure (schéma) générale de la base de données tout en masquant les structures de stockage des données.

4.3 - Niveau interne

Il décrit la structure de stockage physique de la base de données dans un système informatique par des systèmes de gestion de fichiers et des politiques d'accès.

5 - Quelques exemples

	Typologie des SGBD selon la structure de la base de données					
	Hiérarchique	Réseau	Relationnel	Objet	Document	NoSQL
Big Table						Oui
CouchDB					Oui	
DB2			Oui			
DynamoDB					Oui	Oui
Firebird			Oui			
IBM Informix			Oui			

	Typologie des SGBD selon la structure de la base de données					
	Hiérarchique	Réseau	Relationnel	Objet	Document	NoSQL
IDS II		Oui				
IMS	Oui					
InterSystems IRIS			Oui	Oui		
MariaDB			Oui			
Microsoft Access			Oui			
Microsoft SQL Serveur			Oui			
MongoDB					Oui	
MySQL			Oui			
Oracle Database			Oui			
OrientDB					Oui	
PostgreSQL			Oui	Oui		
Sybase ASE			Oui			
Voldemort						Oui

Exercice 06 : **Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{23}(100%)**

SÉQUENCE 3 : MODÉLISATION CONCEPTUELLE DES DONNÉES : MODÈLE ENTITÉ-ASSOCIATION

OBJECTIF 3.1 : RECENSER LES DONNÉES ÉLÉMENTAIRES DU MONDE À MODÉLISER POUR CONSTITUER UN DICTIONNAIRE DE DONNÉES

Durée : 48 heures	Seul de validation : 60%

Le document issu de l'analyse de l'existant et des besoins a permis comprendre le système d'information du monde ou du phénomène à modéliser. De ce document, il faut extraire de multiples données qu'il faut répertorier dans un dictionnaire de données.

1 - Définition du dictionnaire de données

Un dictionnaire de données est un recueil de données élémentaires manipulées dans le système d'information en précisant pour chaque donnée son code ou symbole, sa signification, son type, sa taille et une observation si nécessaire.

Une **donnée élémentaire** est une donnée non décomposable comme nom, prénom, date, prix, âge, poids, longueur, … Comme contre-exemple, ordinateur est une donnée décomposable en capacité de la mémoire, vitesse du processeur, capacité du disque dur, …

Le nom d'une donnée élémentaire peut être assez long et dans ce cas, le concepteur doit le codifier : **code** ou **symbole**. Et comme tout ce qui est codifié n'est pas universellement compréhensible, il faut lui adjoindre une **désignation**. Exemple de code : « VProc » avec pour désignation « vitesse du processeur ».

Le **type de la donnée** est un ensemble de valeurs ou domaine de valeurs que nous avons défini à l'Objectif 1.1 : 2 - Types de données. Bien que la date soit une donnée décomposable en jour, mois et année, elle est considérée en système d'information comme une donnée élémentaire.

La **taille de la donnée** est la limitation de la plage des valeurs. Par exemple pour une donnée de type chaîne de caractères, si taille est fixée à 200 cela veut dire que la donnée aura au plus 200 caractères.

Un **commentaire** est toute information jugée utile pour mieux cerner les contours de la donnée. Ça peut être des contraintes (exemple : l'âge doit être supérieur à 18 ans), la formule de calcul de la dite donnée (exemple : Prix total = Prix unitaire x Quantité), le format de la donnée (exemple : date au format JJ/MM/AAAA),

2 - Epuration du dictionnaire de données

Certaines données doivent enlever du dictionnaire de données

2.1 - Suppression des synonymes

Ce sont des données avec des codes différents mais avec la même sémantique ou signification.

Exemple : Dans un système d'information de gestion du personnel, Matricule et Numéro sont des synonymes.

2.2 - Reformulation des polysèmes

Ce sont des données qui ont le même code mais avec des sémantiques ou significations différentes. Dans ce cas, il faut reformuler l'un des codes ou les deux.

Exemple : Dans un système d'information de gestion de ventes, la donnée « quantité » peut exister 2 fois pour représenter la « quantité commandée » et la « quantité vendue ». Il faudra renommer les 2 en Quantité_Commande et Quantité_Vente.

2.3 - Suppression des redondances

Ce sont des données qui ont le même code et la même signification ou sémantique. Ça peut apparaître souvent lorsqu'on parcoure le document d'analyse pour reporter les données et qu'une donnée apparaisse plus d'une fois dans le document.

2.4 - Suppression des données calculées

Ce sont des données qui sont obtenues par calcul à partir des données existantes.

Exemple : Dans un système d'information de gestion des notes, la moyenne d'un étudiant à une matière est une donnée calculée à base de ses différentes notes de devoirs et d'examens affectées de leur coefficient.

Attention : Plus loin dans la modélisation, vous pourriez voir des données calculées et stockées dans le système d'information ; cela doit être des exceptions et peut se justifier par des soucis d'optimisation car le calcul prend trop de temps.

2.5 - Ajout des données détectées dans les règles de calcul

Parfois, on remarque dans les formules de calcul des données qui ne sont pas dans le dictionnaire de données. Dans ce cas, il faut les ajouter.

Exemple : Dans un système d'information de gestion de vente, on se rend compte qu'à chaque produit, il faut ajouter 18% du prix d'achat sur le prix de vente. Cela veut dire que la TVA est de 18% qu'il faudra insérer dans le dictionnaire de données sachant qu'elle peut changer si la législation venait à changer. Or si elle était figée dans le système, il faudrait reprendre le système

2.6 - Décomposition des données concaténées ou non élémentaires

Ce sont des données contenant plusieurs données existantes ou pas dans le dictionnaire de données.

Exemple : Adresse est une donnée concaténée qu'il faut décomposer en Rue, Ville, Pays, Code postal, Boite postale, …

3 - Formalisme du dictionnaire de données

Le dictionnaire de données est formalisé par un tableau à 5 colonnes et autant de lignes que de données à y inscrire.

Code	Signification	Type	Taille	Observation

4 - Exemple de dictionnaire de données du système d'information de gestion d'inscription des étudiants dans une université

Code	Signification	Type	Taille	Observation
INE	Numéro d'Identification Nationale de l'Etudiant	Entier		
NomE	Nom de l'étudiant	Chaîne de caractères	150	
DateInscription	Date d'inscription	Date		JJ/MM/AAAA
NomF	Nom de la formation ou de la filière	Chaîne de caractères	50	

A vous de compléter ce dictionnaire de données !

5 - Exemple d'application : e-Commerce

Code	Signification	Type	Taille	Observation
Adresse	Adresse du fournisseur	Chaîne de caractères	200	
CodeCat	Code de la catégorie des produits	Chaîne de caractères	20	
CodeP	Code du produit	Chaîne de caractères	20	
CodeR	Code du mode de règlement de la facture	Chaîne de caractères	20	
DateA	Date de l'approvisionnement	Date		JJ/MM/AAAA
DateCde	Date de la commande	Entier		JJ/MM/AAAA
DateF	Date de la facture	Date		JJ/MM/AAAA
DateNaissane	Date de naissance du client	Date		JJ/MM/AAAA
Désignation	Nom du fournisseur	Chaîne de caractères	150	
EmailC	Adresse électronique du client	Chaîne de caractères	70	
EmailF	Adresse électronique du fournisseur	Chaîne de caractères	70	
Identifiant	Identifiant de l'approvisionnement	Entier		
LibelléCat	Nom de la catégorie des produits	Chaîne de caractères	100	
LibelléP	Nom du produit	Chaîne de caractères	100	
LibelléR	Nom du mode de règlement de la facture	Chaîne de caractères	100	
Login	Nom d'utilisateur du client pour se connecter au site	Chaîne de caractères	30	
NINEA	Numéro d'Identification National des Entreprises et des Associations	Entier	15	
Nom	Nom du client	Chaîne de caractères	100	
NuméroCde	Numéro de la commande	Chaîne de caractères	20	

Code	Signification	Type	Taille	Observation
NuméroF	Numéro de la facture	Chaîne de caractères	20	
Password	Mot de passe du client	Chaîne de caractères	25	La longueur doit être supérieure ou égale à 8
Prénom	Prénom du client	Chaîne de caractères	200	
PrixUnitaire	Prix unitaire de vente du produt	Entier		
Quantité	Quantité de produits approvisionnés	Entier		
QuantitéCde	Quantité de produits commandés	Entier		
QuantitéLivrée	Quantité de produits livrés	Entier		
QuantitéStock	Quantité en stock du produit	Entier		
Référence	Référence du client	Chaîne de caractères	20	
SeuilAlerte	Seuil d'alerte du produit	Entier		
Sexe	Sexe du client	Caractère		
SiteWeb	Site web du fournisseur	Chaîne de caractères	150	
TéléphoneC	Téléphone du client	Chaîne de caractères	10	Le numéro peut contenir l'indicateur téléphonique du pays
TéléphoneF	Téléphone du fournisseur	Chaîne de caractères	10	Le numéro peut contenir l'indicateur téléphonique du pays

Exercice 07 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{31}(20%)

Exercice 08 : Seuil de validation : 60% Titre : Dictionnaire de données de la gestion d'un institut de formation Objectifs visés : O_{31}(80%)

Exercice 09 : Seuil de validation : 60% Titre : Dictionnaire de données de la gestion des états civils Objectifs visés : O_{31}(80%)

Exercice 10 : Seuil de validation : 60% Titre : Dictionnaire de données de la gestion d'une chaîne hôtelière Objectifs visés : O_{31}(80%)

OBJECTIF 3.2 : CONSTITUER LES ENTITÉS À PARTIR DES DONNÉES ÉLÉMENTAIRES

Durée : 48 heures	Seul de validation : 60%

Le dictionnaire de données décrit toutes les données élémentaires manipulées dans le système d'information. Or dans le monde réel les données manipulées au premier plan sont rarement élémentaires. Par exemple, dans un système d'information de gestion d'une université, on vous parlera d'étudiants, formations, unités d'enseignement, enseignants, tuteurs, filière, facultés, pôles, unité de formation et de recherche, … Toutes ces informations sont composées des données élémentaires contenues dans le dictionnaire de données.

1 - Entité

Une entité est la représentation d'un élément matériel (concret) ou immatériel (abstrait) ayant un rôle dans le monde ou phénomène modélisé. Les entités sont des macro-données du système d'information comme étudiant, enseignant, tuteur, cours, formation, …

1.1 - Propriétés d'entité

Une entité n'est pas une donnée élémentaire et est caractérisée par des données élémentaires qu'on appelle **propriété** provenant du dictionnaire de données.

Une propriété permet de caractériser une entité et appartient seulement à cette entité.

Une entité possède un certain nombre de propriétés pour la caractériser.

Exemples :

- L'entité ***riz local*** est caractérisée par les propriétés : R76, riz, 4500, 23, 25/11/2025, …
- L'entité ***Loi sur le LMD*** est caractérisée par les propriétés : TS035, 12/09/2007, Loi d'organisation du LMD dans l'enseignement supérieur, "contenu de la loi", Abdoulaye WADE, …
- L'entité ***Algorithmique*** est caractérisée par les propriétés LSIMAC1121, Algorithmique, 6 et 3.

1.2 - Entité-type ou type d'entité

Une entité-type est un ensemble d'entités qui possèdent une sémantique et des propriétés communes.

Exemples :

- L'entité-type ***produit*** caractérisée par les propriétés numéro, nom, prix unitaire, quantité en stock, date de péremption, …
- L'entité-type ***loi*** caractérisée par les propriétés numéro, date de promulgation, objet, contenu, signataire, …
- L'entité-type ***unité d'enseignement*** caractérisée par les propriétés code, libellé, crédit et coefficient.

Attention : Par abus de langage et ça sera le cas dans le cadre de ce cours, l'entité-type est appelé entité pour simplifier l'appel et l'écriture.

1.3 - Formalisme

a) Selon le modèle entité-association défini par Peter Chen

Le modèle entité-association créé par Peter Chen dans les années 70 propose de représenter l'entité-type dans un rectangle et la propriété dans une ellipse. Chaque propriété d'une entité est formalisée par un trait le liant à son entité.

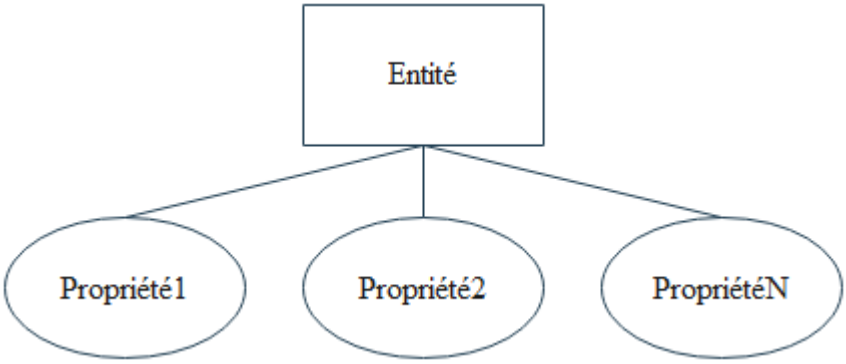


Figure 5 : Formalisme d'une entité-type selon Peter Chen

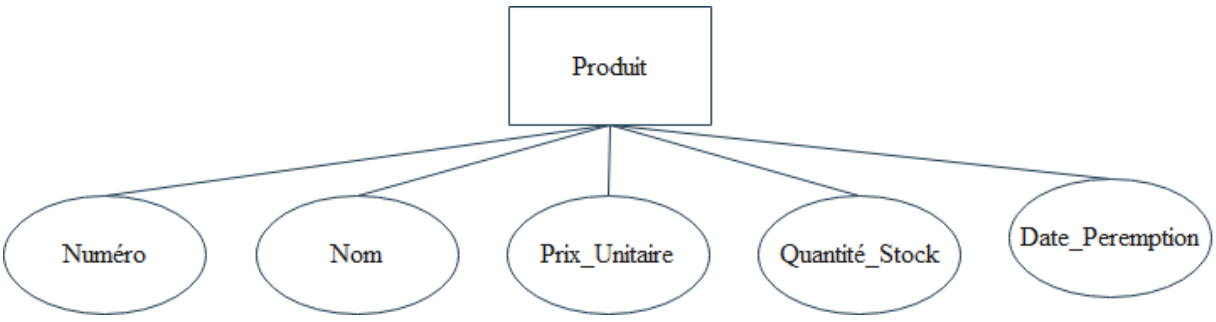


Figure 6 : Exemple d'une entité-type selon Peter Chen

b) Selon la méthode MERISE

Dans les années 80, la méthode MERISE a été largement utilisée pour modéliser les systèmes d'information tant du point de vue des données que des traitements.

Une entité-type est formalisée par un rectangle à 2 parties avec la partie supérieure (entête) contenant le nom de l'entité-type et la partie inférieure (corps) la liste des propriétés, chacune sur une ligne.

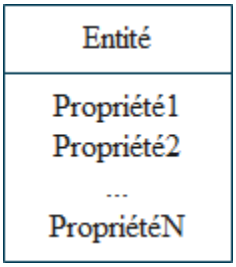


Figure 7 : Formalisme d'une entité-type selon la méthode MERISE

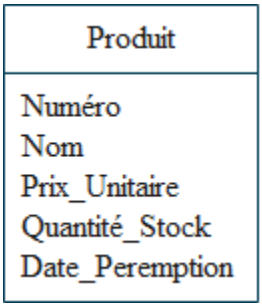


Figure 8 : Exemple d'une entité-type selon MERISE

1.4 - Occurrence d'entité-type

Une occurrence d'une entité-type est en réalité l'entité définit ci-dessus et est définie par une valeur pour chacune de ses propriétés.

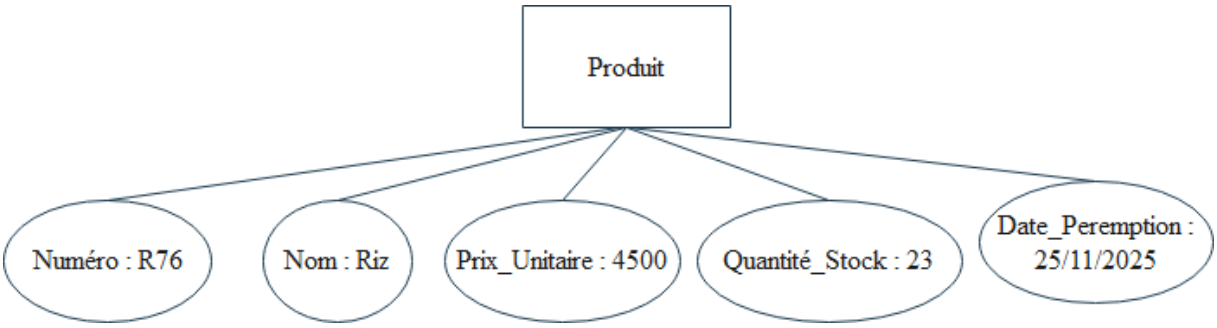


Figure 9 : Occurrence de l'entité-type produit selon Peter Chen

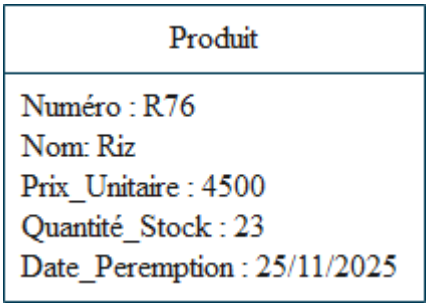


Figure 10 : Occurrence de l'entité-type produit selon MERISE

2 - Dépendance fonctionnelle

2.1 - Définition

Une dépendance fonctionnelle entre 2 propriétés A et B et notée **A → B** est un lien fonctionnel qui décrit la dépendance de B à A.

Une propriété B dépend fonctionnellement de la propriété A et notée **A → B** si et seulement si à une valeur de A, il n'est possible d'associer qu'une et une seule valeur de B ; la réciproque est fausse.

Exemples :

- INE → Nom_Etudiant
- Numero_Extrait_Naissance → Date_Naissance

La partie gauche d'une dépendance fonctionnelle est appelée **source** et la partie droite est appelée **but**.

Une dépendance fonctionnelle dont la source est composée d'une seule propriété est appelée **dépendance fonctionnelle simple**.

Une dépendance fonctionnelle dont la source est composée de plusieurs propriétés est appelée **dépendance fonctionnelle composée**.

Exemple : Matricule_Etudiant, Code_Matière → Note

Une **dépendance fonctionnelle directe** est une dépendance fonctionnelle dont le but dépend de la source sans autre propriété intermédiaire. On dira également qu'une dépendance fonctionnelle A → B est directe s'il n'existe pas de propriété C tel que les dépendances fonctionnelles A → C et C → B soient correctes.

Exemple de dépendance fonctionnelle indirecte : Numero_Facture → Nom_Magasin. Elle est indirecte car Numero_Facture→Numero_Magasin et Numero_Magasin→Nom_Magasin.

Une **dépendance fonctionnelle élémentaire** est une dépendance fonctionnelle dont la source ne contient pas de propriété superflue c'est-à-dire, en l'éliminant, la dépendance reste correcte. On dira également qu'une dépendance fonctionnelle A, B → C est non élémentaire si A → C est une dépendance correcte ; dans ce cas, B est une propriété superflue.

2.2 - Axiomes d'Armstrong

Les axiomes d'Armstrong sont des propriétés mathématiques que respectent les dépendances fonctionnelles.

a) Réflexivité

Tout attribut se détermine lui-même : X → X

Tout groupe d'attributs qui se détermine lui-même : X,Y → X,Y

Tout groupe d'attributs détermine chacun de ses attributs ou partie de ses attributs : X,Y,Z → X et X,Y,Z → Y et X,Y,Z → Z et X,Y,Z → X,Y et X,Y,Z → X,Z et X,Y,Z → Y,Z

b) Augmentation

Si X → Y alors X,Z → Y,Z

X, Y et Z sont des attributs ou des groupes d'attributs

c) Transitivité

Si X→ Y et Y→ Z alors X → Z

X, Y et Z sont des attributs ou des groupes d'attributs

Une dépendance fonctionnelle transitive est une dépendance fonctionnelle indirecte.

2.3 - Autres propriétés déduites des axiomes d'Armstrong

a) Union

Si X → Y et X → Z alors X → X,Z

Cette propriété est déduite de la réflexivité, de l'augmentation et de la transitivité

b) Pseudo-Transitivité

Si X → Y et WY → Z alors X,W → Z

Cette propriété est déduite de l'augmentation et de la réflexivité

c) Décomposition

Si X → Y,Z alors X → Y et X → Z

Cette propriété est déduite de la réflexivité et de la transitivité

2.4 - Graphe de dépendances fonctionnelles

Un graphe de dépendances fonctionnelles est un graphe contenant l'ensemble des dépendances fonctionnelles entre les propriétés tout en éliminant les dépendances fonctionnelles indirectes ou obtenues par transitivité.

3 - Identifiant d'entité

Un identifiant d'entité est une propriété qui a une valeur unique pour chaque occurrence de cette entité.

Exemples :

- Le matricule est une valeur unique pour un étudiant dont c'est un identifiant
- L'adresse électronique est propre à chaque étudiant, de même que son téléphone ; donc email et téléphone sont des identifiants.

Contre-exemple : Le nom de l'étudiant n'est pas unique à un seul étudiant ; donc il ne peut pas être un identifiant.

Le Graphe des Dépendances Fonctionnelles est un bon outil pour déterminer les identifiants

3.1 - Identifiant candidat

Un identifiant candidat est une propriété d'une entité susceptible de la représenter de façon unique. Une entité a forcément un identifiant candidat et peut en avoir plusieurs.

Exemple : Identification nationale de l'Etudiant, Email, Téléphone sont des identifiants candidats de l'entité Etudiant.

3.2 - Identifiant primaire

Un identifiant primaire d'une entité est un identifiant candidat choisi par le concepteur de la base de données pour représenter de façon unique l'entité.

Un identifiant primaire d'une entité est une propriété dont dépend toutes les autres propriétés de l'entité.

Attention : Par abus de langage, on utilise le terme identifiant pour faire allusion à l'identifiant primaire

3.3 - Formalisme

Un identifiant primaire est souligné.

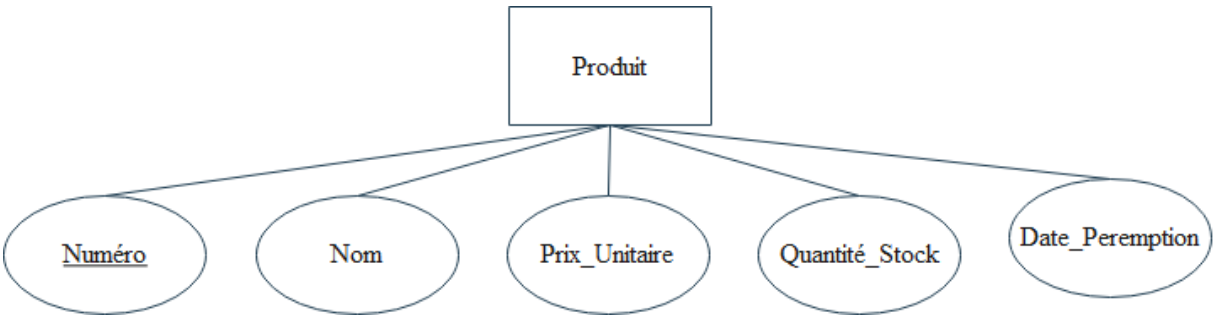


Figure 11 : Exemple d'une entité avec identifiant selon Peter Chen

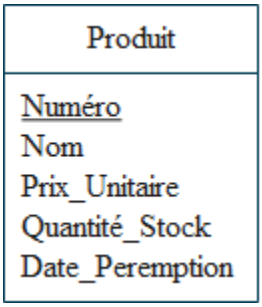


Figure 12 : Exemple d'une entité avec identifiant selon MERISE

Remarque : Il peut arriver que dans le dictionnaire de données, aucune propriété ne puisse permettre l'identification d'une entité de façon unique. 2 cas de figure se présente :

- Il est possible que la combinaison de plusieurs propriétés d'une entité représente son identifiant
- Si aucune combinaison de propriétés ne puisse représenter l'identifiant de l'entité, il faudra créer une propriété qui sera l'identifiant.

4 - Exemple d'application : e-Commerce

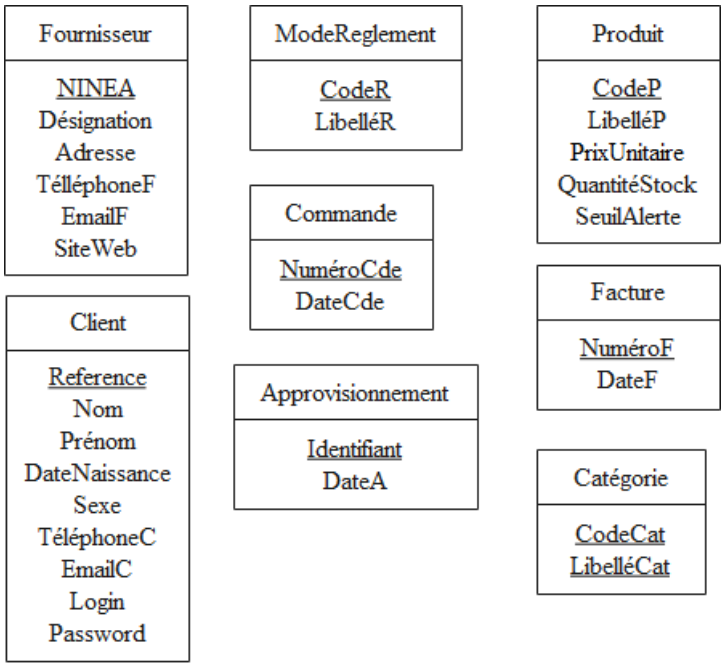


Figure 13 : Liste des entités du site d'e-commerce représentées selon la méthode MERISE

Exercice 11 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{32}(20%)

Exercice 12 : Seuil de validation : 60% Titre : Entités de la gestion d'un institut de formation Objectifs visés : O_{31}(80%)

Exercice 13 : Seuil de validation : 60% Titre : Entités de la gestion des états civils Objectifs visés : O_{31}(80%)

Exercice 14 : Seuil de validation : 60% Titre : Entités de la gestion d'une chaîne hôtelière Objectifs visés : O_{31}(80%)

OBJECTIF 3.3 : METTRE LES ENTITÉS EN RELATION POUR CONSTITUER LE MODÈLE ENTITÉ-ASSOCIATION

Durée : 72 heures	Seul de validation : 75%

Les entités du monde ou du phénomène à modéliser ne sont pas des objets isolés auquel cas ça ne sera plus un système ! Elles sont en relation entre-elles en fonction de leur utilité dans le système.

1 - Association

1.1 - Définition

Une association est une relation ou lien sémantique entre deux ou plusieurs entités. Elle est définie par le concepteur de la base de données en fonction de sa compréhension du problème posé.

Une **association-type** ou **type d'association** est un ensemble d'associations qui possèdent une sémantique et des propriétés communes.

Comme dans le cas des entités, l'association est une occurrence pour une association-type ou type d'association.

1.2 - Caractéristiques

Une association possède :

- Des propriétés si nécessaire
- Des rôles si nécessaire pour certaines précisions
- Des cardinalités entre parenthèses

1.3 - Formalisme

Le nom de l'association est un verbe à l'infinitif.

a) Selon le modèle entité-association défini par Peter Chen

L'association est représentée par un losange contenant le nom de l'association. Le lien est représenté par un trait entre l'association et l'entité dont figurera les cardinalités et éventuellement le nom de rôles.

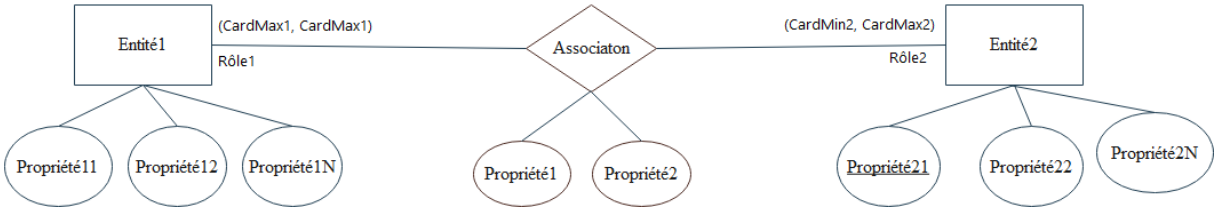


Figure 14 : Formalisme d'une association-type selon Peter Chen

b) Selon le modèle MERISE

L'association est représentée par une ellipse à 2 parties dont un entête (partie supérieure) contenant le nom de l'association et un corps (partie inférieure) contenant ses propriétés. Le lien est représenté par un trait entre l'association et l'entité dont figurera les cardinalités et éventuellement le nom de rôles.

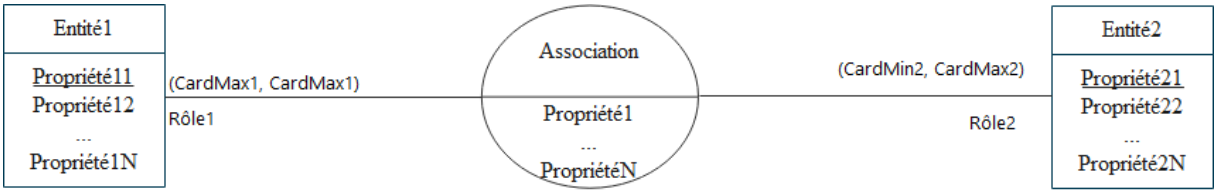


Figure 15 : Formalisme d'une association-type selon la méthode MERISE

1.4 - Dimension ou degré ou arité d'une association

La dimension ou degré ou arité d'une association est le nombre d'entités participant à l'association.

2 - Détermination des associations

Elle peut se faire de façon intuitive et par expérience en ayant bien compris le problème posé.

Exemple : Soit un système d'information de vente de produits. De façon intuitive et par expérience, je sais qu'il y aura une association nommée « acheter » entre les entités Produit et Client.

La façon la plus sûre est l'utilisation du Graphe de Dépendances Fonctionnelles.

2.1 - Dépendance fonctionnelle directe entre identifiants primaires

Lorsqu'il y'a une dépendance fonctionnelle directe entre les identifiants primaires de 2 entités cela veut dire que ces 2 entités sont en association sans propriété.

C'est au concepteur de trouver le verbe sui va le mieux pour exprimer l'association.

Exemple : Soit les dépendances fonctionnelles suivantes

Matricule → Nom, Prénom, DateNaissance

CodeF → NomF

Matricule → CodeF

Leur interprétation donne l'association « Inscrire » représentée ainsi :

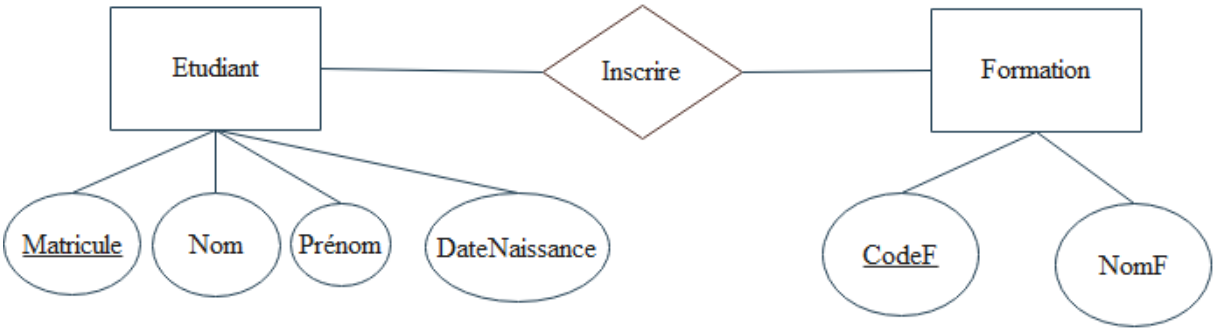


Figure 16 : Exemple d'association déterminée à partir une dépendance fonctionnelle entre les identifiants de 2 entités et représentée selon Peter Chen

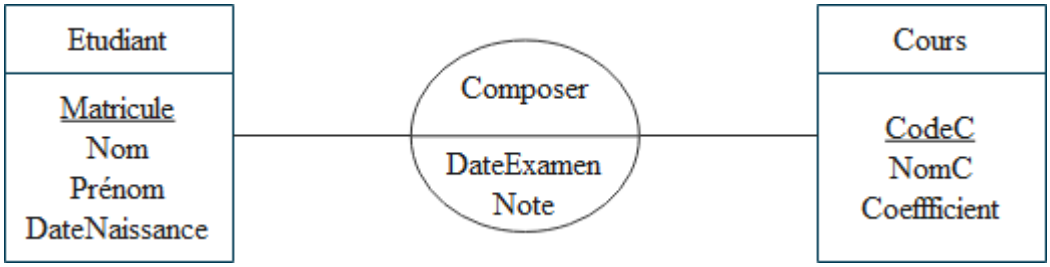


Figure 17 : Exemple d'association déterminée à partir une dépendance fonctionnelle entre les identifiants de 2 entités et représentée la méthode MERISE

2.2 - Dépendance fonctionnelle directe avec une source composée des identifiants primaires

Lorsque la source d'une dépendance fonctionnelle est composée des identifiants primaires des entités cela veut dire que ces entités sont en association et les propriétés de l'association sont le but de la dépendance fonctionnelle.

C'est au concepteur de trouver le verbe sui va le mieux pour exprimer l'association.

Exemple : Soit les dépendances fonctionnelles suivantes

Matricule → Nom, Prénom, DateNaissance

CodeC → NomC, Coefficient

Matricule, CodeC → DateExamen, Note

Leur interprétation donne l'association « Inscrire » représentée ainsi :

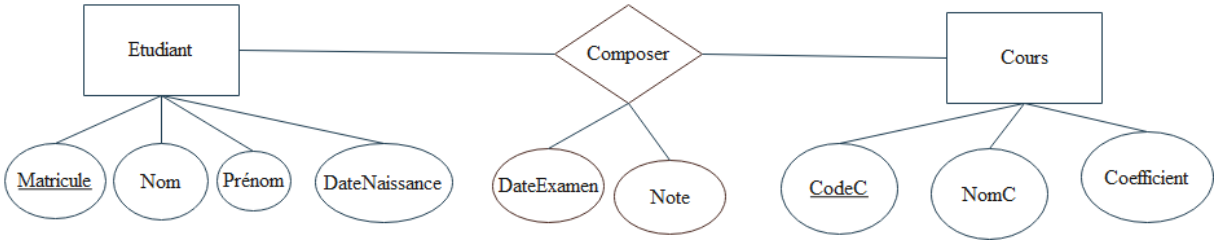


Figure 18 : Exemple d'association déterminée à partir une dépendance fonctionnelle avec une source composée des identifiants primaires et représentée selon Peter Chen

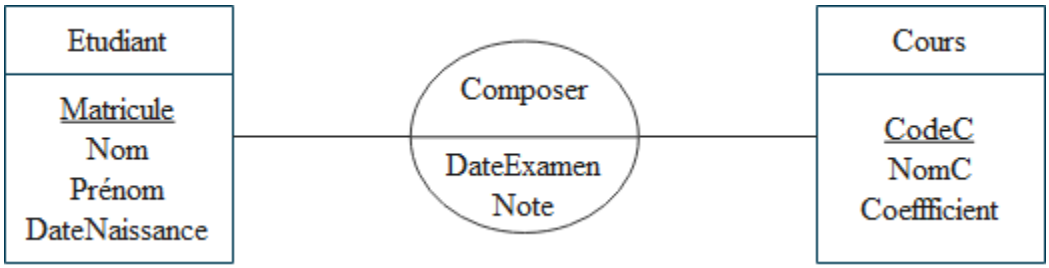


Figure 19 : Exemple d'association déterminée à partir une dépendance fonctionnelle avec une source composée des identifiants primaires et représentée selon le modèle MERISE

3 - Cardinalité d'une association

3.1 - Définition

La cardinalité d'une association est le nombre minimal et maximal de fois qu'une occurrence d'une entité associée est en liaison avec l'autre entité. Une association possède une paire de cardinalités (Cardinalité minimale et cardinalité maximale) à chacune de ses extrémités (portées par les entités) et décrit le nombre de participation de cette entité à l'association.

La cardinalité minimale (généralement 0 ou 1) est le nombre minimal de fois que l'entité doit participer à l'association

La cardinalité maximale (généralement 0, 1 ou n) est le nombre maximal de fois que l'entité doit participer à l'association

3.2 - Détermination des cardinalités

Les cardinalités sont déterminées à partir des règles de gestion énoncées dans le document d'analyse.

Exemple 1 : Un étudiant s'inscrit dans une seule formation.

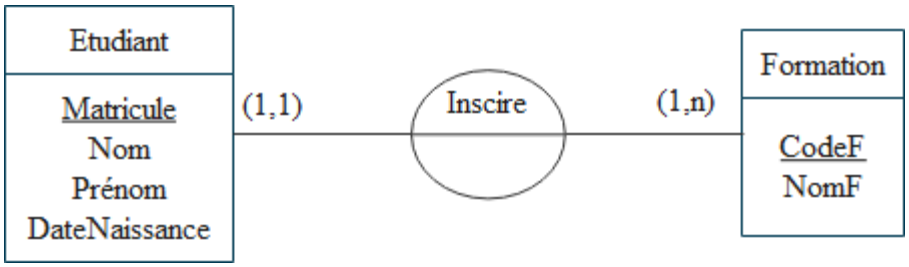


Figure 20 : Exemple 1 de cardinalités représentées selon la méthode MERISE

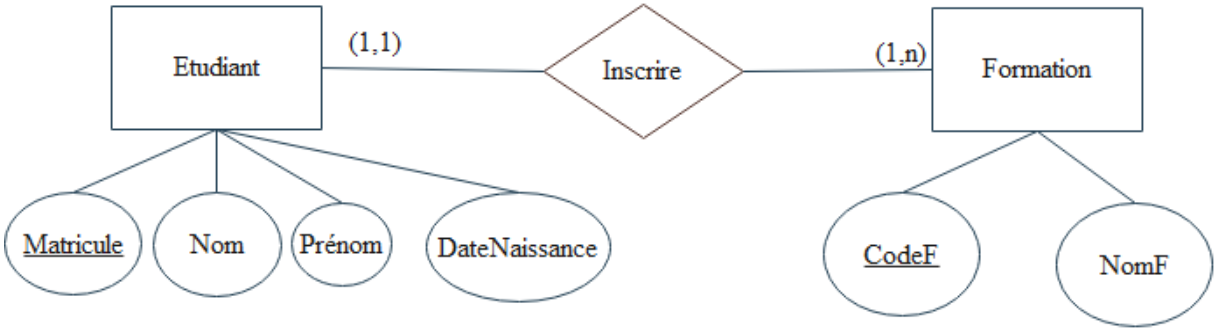


Figure 21 : Exemple 1 de cardinalités représentées selon la Peter Chen

Cette association se lit ainsi :

- Un étudiant s'inscrit au minimum à 1 et au maximum à 1 formation. Autrement dit, un étudiant s'inscrit dans une et une seule formation
- Une formation contient au minimum 1 et au maximum n (plusieurs) étudiants

Exemple 2 : Un étudiant fait de plusieurs examens pour un cours à des dates différentes.

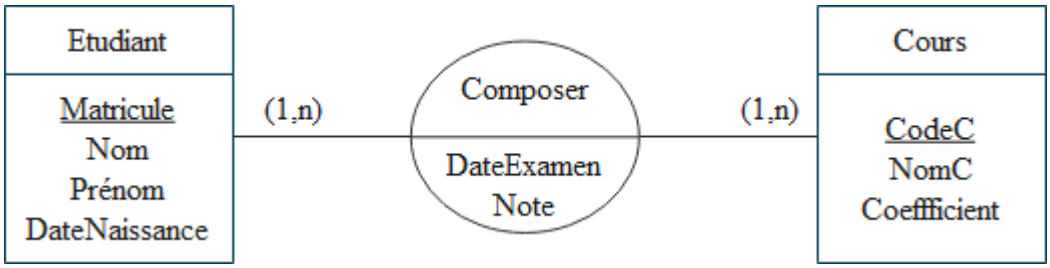


Figure 22 : Exemple 2 de cardinalités représentées selon la méthode MERISE

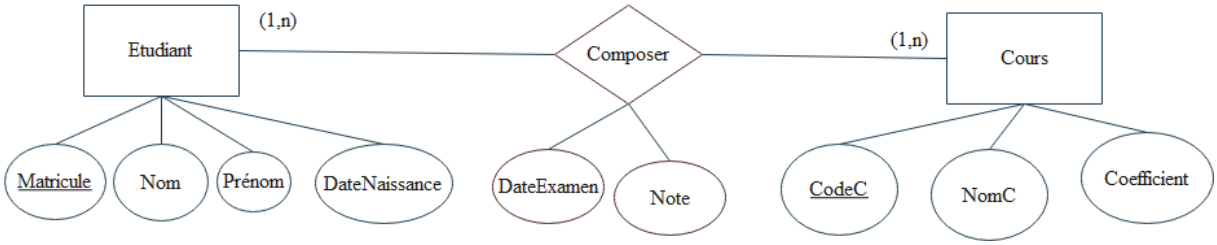


Figure 23 : Exemple 2 de cardinalités représentées selon Peter Chen

Cette association se lit ainsi :

- Un étudiant compose au minimum à 1 et au maximum dans plusieurs cours
- Un cours est composé au minimum par 1 et au maximum par plusieurs étudiants

Si par hasard, le système prévoit que les étudiants ne puissent pas composer à l'ensemble des examens d'un cours, il faudra mettre la cardinalité minimale du côté étudiant à 0.

De même si un cours peut ne jamais subir des évaluations, il faudra également mettre à 0 la cardinalité minimale du côté du cours.

3.3 - Typologie des cardinalités

Soient les associations représentées dans le formalisme à la Figure 14 : Formalisme d'une association-type selon Peter Chen et la Figure 15 : Formalisme d'une association-type selon la méthode MERISE.

a) Cardinalité « père-fils »

Une cardinalité « père-fils » est une cardinalité dont les valeurs potentielles sont les suivantes :

CardMin1	CardMax1		CardMin2	CardMax2
1	1		1	N
1	1		0	N
0	1		1	N
0	1		0	N

b) Cardinalité plusieurs à plusieurs

Une cardinalité « plusieurs à plusieurs » est une cardinalité dont les valeurs potentielles sont les suivantes :

CardMin1	CardMax1		CardMin2	CardMax2
1	N		1	N
1	N		0	N
0	N		1	N
0	N		0	N

a) Cardinalité « au plus un »

Une cardinalité « au plus un » est une cardinalité dont les valeurs potentielles sont les suivantes :

CardMin1	CardMax1		CardMin2	CardMax2
1	1		0	1
0	1		1	1
0	1		0	1

b) Cardinalité « un à un »

Une cardinalité « un à un » est une cardinalité dont les valeurs potentielles sont les suivantes :

CardMin1	CardMax1		CardMin2	CardMax2
1	1		1	1

4 - Typologie des associations

4.1 - Association réflexive

Une association est réflexive lorsque l'entité est en relation avec elle-même. Il est important de donner les noms de rôles de l'association pour mieux la lire.

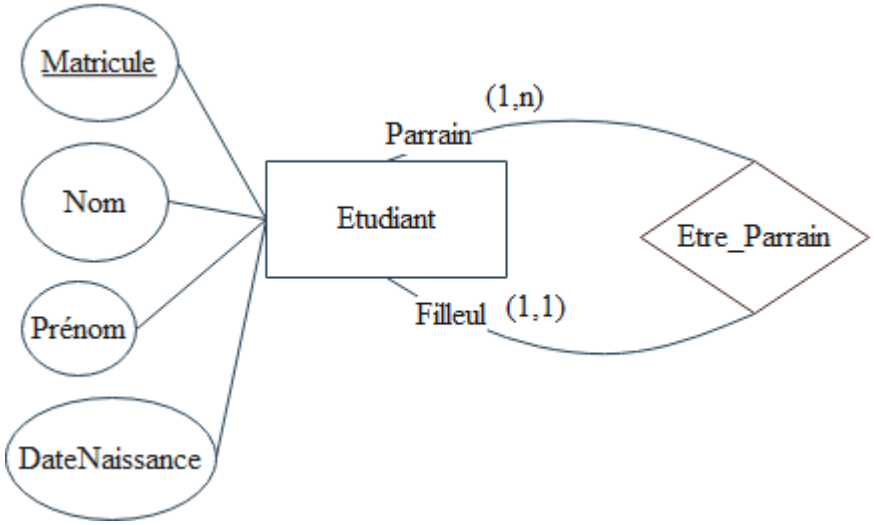


Figure 24 : Exemple d'association réflexive représentée selon Peter Chen

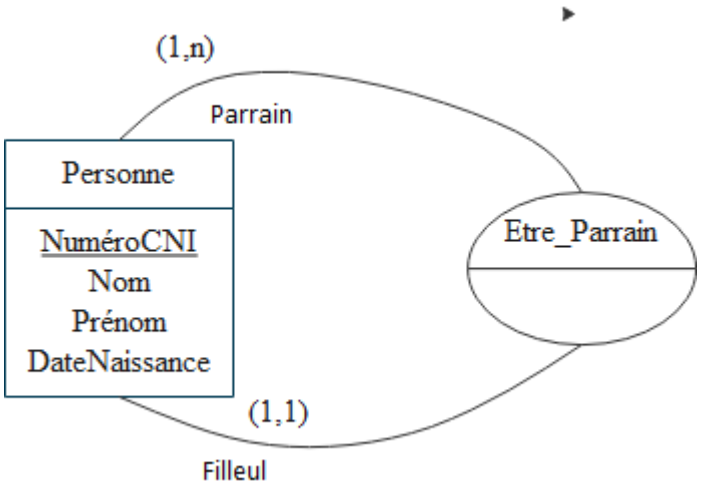


Figure 25 : Exemple d'association réflexive représentée selon la méthode MERISE

L'association se lit ainsi :

- Un parrain parraine 1 ou plusieurs filleuls
- Un filleul est parrainé par un et un seul parrain

4.2 - Association n-aire

Une association n-aire est une association avec n (n≥2) entités.

Lorsque n = 2, on parle d'**association binaire**

Lorsque n = 3, on parle d'**association ternaire**

Lorsque n > 3, on parle d'**association n-aire**

4.3 - Autres associations issues de la méthode MERISE 2

La seconde version de MERISE étend la première version en introduisant des concepts permettant de mieux exprimer les relations entre les entités.

a) Lien spécialisation ou de généralisation

Certaines entités sémantiquement proches peuvent avoir certaines propriétés identiques. Dans ce cas, on créé une entité dont ses propriétés seront les propriétés communes aux entités. Cette nouvelle entité appelée « **entité-mère** » est reliée aux autres (**entités-filles**) par un lien de spécialisation et il faut supprimer des entités-filles les propriétés appartenant à l'entité-mère.

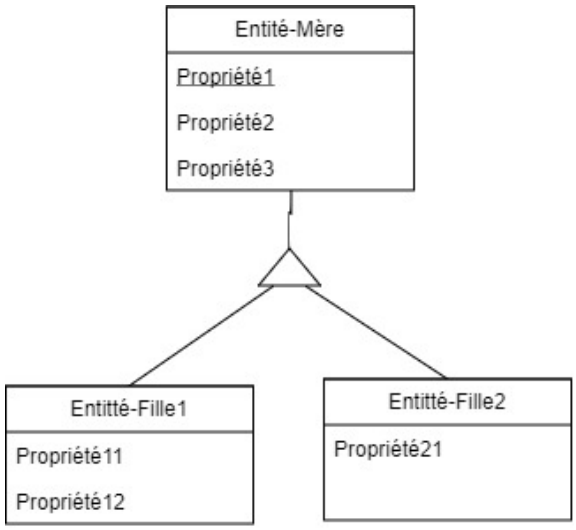


Figure 26 : Formalisme du lien de spécialisation ou généralisation

La **spécialisation** est l'extraction des propriétés communes de plusieurs entités (entités-filles) sémantiquement proches pour en constituer une nouvelle entité (entité-mère) qu'on reliera avec les autres entités-filles.

b) Lien d'héritage

Durant la modélisation, il peut arriver qu'une nouvelle entité E1 à ajouter au modèle ait des propriétés incluant toutes les propriétés d'une autre entité E2. Dans ce cas, on considère l'entité E2 comme **entité-mère** et pas besoin de créer cette nouvelle entité E2 avec les propriétés appartenant déjà à entité E1. Il faut juste créer la nouvelle entité E2 avec ses propriétés spécifiques n'appartenant pas à l'entité E1 et les relier par un lien d'héritage. Le formalisme du lien d'héritage est identique à celui de la spécialisation ou généralisation.

L'**héritage** est le fait de créer une entité-fille avec moins de propriétés (propriétés spécifique) qui bénéficie des propriétés appartenant à l'entité-mère par un lien entre les deux.

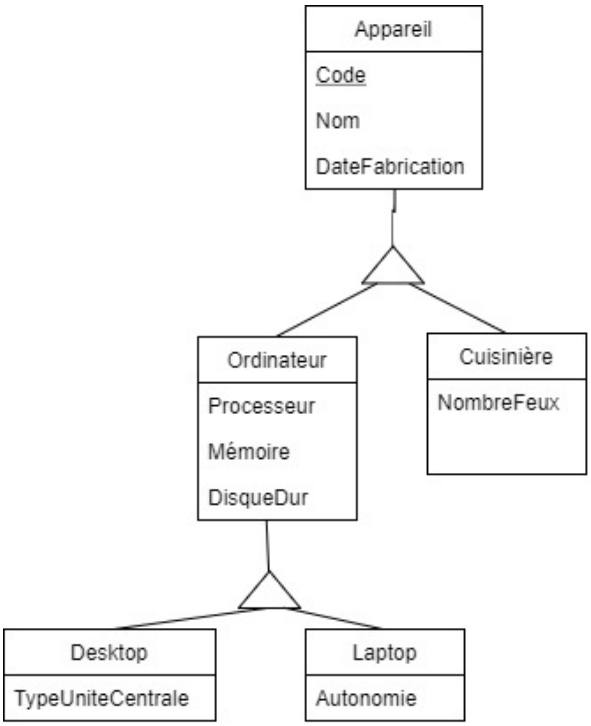


Figure 27 : Exemple de représentation des liens d'héritage et spécialisation

Remarque : Les liens d'héritage et de spécialisation ne porte pas de cardinalité.

5 - Contraintes d'intégrité

Le monde réel ou le phénomène à modéliser obéit à des règles de gestion figurant dans le document d'analyse. De même votre base de données doit respecter ses règles pour garantir la représentation du monde réel ou du phénomène modélisé. De plus, il y a des règles découlant du modèle entité-association (théorie des bases de données) à respecter. Toutes ces règles sont appelées les **contraintes d'intégrité** et concourent à créer une base de données respectant ses objectifs notamment la structuration, la cohérence et la non-redondance des données.

Une **contrainte d'intégrité** est toute règle implicite ou explicite que doivent respecter les données.

5.1 - Contrainte d'entité

- Toute entité doit avoir au moins une propriété
- Deux entités différentes ne peuvent pas avoir le même nom
- Deux propriétés d'une même entité ne peuvent pas avoir le même nom

5.2 - Contrainte de domaine

Chaque propriété doit avoir un domaine de valeurs et les occurrences de cette propriété doivent appartenir à ce domaine.

5.3 - Contrainte d'unicité

Toute entité doit avoir un identifiant unique et son occurrence est aussi unique.

5.4 - Contrainte d'intégrité référentielle

Certaines données ne peuvent exister sans d'autres ; cela se traduit par des associations entre les entités. Par exemple, dans une association « composer » de type plusieurs à plusieurs représentant l'évaluation entre une entité « Etudiant » et une entité « Matière », on ne doit pas trouver dans cette association un étudiant ayant composé alors qu'il n'existe pas dans l'entité « Etudiant » ; il en est de même pour l'entité « Matière ».

5.5 - Autres contraintes

Il existe des contraintes difficilement exprimables au niveau du modèle entité-association. Il s'agit en général de certaines règles logiques et de gestion. Par exemple, on ne peut vendre un produit dont la quantité en stock est inférieure à la quantité demandée.

6 - Quelques logiciels pour concevoir le modèle entité-association

- **Draw.io** : https://draw-io.fr.softonic.com/telecharger
- **Wondershare EdrawMax** : https://www.edrawsoft.com/download.html

7 - Exemple d'application : e-Commerce

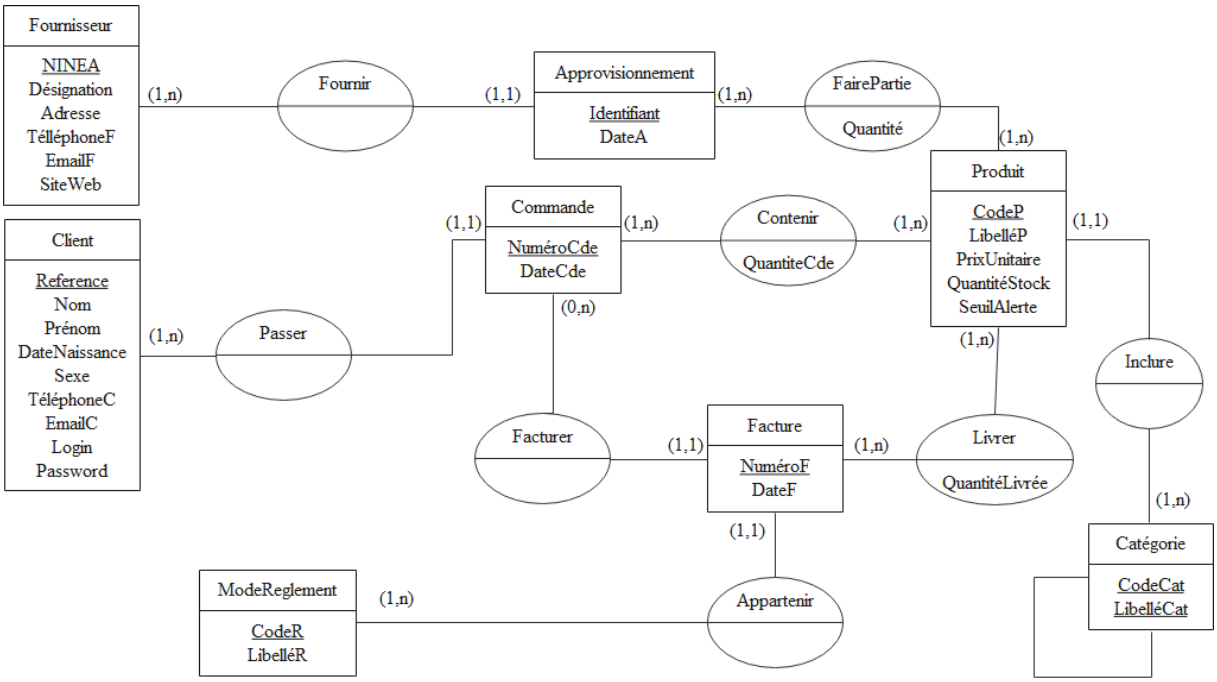


Figure 28 : Modèle Entité-Association du site d'e-commerce représenté selon la méthode MERISE

Exercice 15 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{33}(20%)

Exercice 16 : **Seuil de validation : 60% Titre : Modèle Entité-Association de la gestion d'un institut de formation Objectifs visés : O_{31}(80%)**

Exercice 17 : **Seuil de validation : 60% Titre : Modèle Entité-Association de la gestion des états civils Objectifs visés : O_{31}(80%)**

Exercice 18 : **Seuil de validation : 60% Titre : Modèle Entité-Association de la gestion d'une chaîne hôtelière Objectifs visés : O_{31}(80%)**

SÉQUENCE 4 : MODÉLISATION LOGIQUE DES DONNÉS : MODÈLE RELATIONNEL

OBJECTIF 4.1 : DÉCRIRE LES PRINCIPES DU MODÈLE RELATIONNEL

Durée : 6 heures	Seul de validation : 60%

1 - Définition

Le modèle relationnel est inventé par Edgar Frank Codd dans les années 70. Il est le premier modèle de bases de données sans critère de stockage. Le modèle relationnel est un ensemble de relations basées sur la théorie des ensembles en mathématique qui modélisent les relations existantes entre les données ou informations. C'est le modèle le plus utilisé par de nombreux SGBD qui puise son origine dans la notion de relation.

Mathématiquement, une relation (binaire) entre deux ensembles A et B est un sous-ensemble du produit cartésien de A et B. Le produit cartésien de A et B noté **AxB** est l'ensemble de toutes les paires possibles constituées d'un élément de A et d'un élément de B.

Une relation est une notion abstraite que l'on peut représenter de plusieurs manières.

Soient les ensembles Code = {1, 2, 3, 4, 5} et Nom = {X, Y, Z}. On définit une relation entre les 2 ensembles telle que

- 1 soit en relation avec X et Z
- 3 soit en relation avec X
- 4 soit en relation avec Y

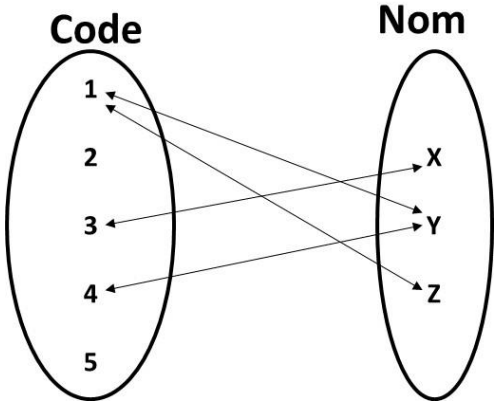


Figure 29 : Représentation par un graphe d'une relation entre 2 ensembles

Code	Nom
1	Y
1	Z
3	X
4	Y

Figure 30 : Représentation sous forme de tableau d'une relation entre 2 ensembles

Le produit cartésien étant un opérateur n-aire permet la généralisation de la notion de relation entre plusieurs ensembles.

Dans une base de données, on utilise toujours la représentation sous forme de tableau.

2 - Quelques opérations sur les ensembles

Soient les ensembles A = {X, P} et B = {X, Y, Z}.

2.1 - Union

L'union de A et B noté **A∪B** est l'ensemble des éléments de A et B

A∪B = {X, P, Y, Z}

2.2 - Intersection

L'intersection de A et B noté **A∩B** est l'ensemble des éléments appartenant à A et à B

A∩B = {X}

2.3 - Produit cartésien

Le produit cartésien de A et B noté **AxB** est l'ensemble de toutes les paires possibles constituées d'un élément de A et d'un élément de B

AxB = {(X, X), (X, Y), (X, P), (P, X), (P, Y), (P, Z)}

2.4 - Différence

La différence entre A et B noté **A-B** est l'ensemble des éléments qui appartiennent à A et pas à B

A-B = {P}

3 - Objectifs

- Fournir une approche méthodologique dans la construction des schémas de données
- Proposer des schémas de données faciles à utiliser

- Permettre un haut degré d'indépendance des programmes d'applications et des activités interactives à la représentation interne des données, en particulier aux choix des ordres d'implantation des données dans les fichiers, des index et plus généralement des chemins d'accès
- Mettre à la disposition des utilisateurs des langages de haut niveau
- Permettre le développement de langages de manipulation de données non procéduraux basés sur des théories solides
- Optimiser les accès à la base de données
- Fournir une base solide pour traiter les problèmes de cohérence et de redondance des données
- Être un modèle extensible permettant de modéliser et de manipuler simplement des données tabulaires, mais pouvant être étendu pour modéliser et manipuler des données complexes
- Devenir un standard pour la description et la manipulation des bases de données

Exercice 19 : **Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{41}(100%)**

OBJECTIF 4.2 : DÉCRIRE LA NOTION DE RELATION

Durée : 6 heures	Seul de validation : 50%

1 - Définition

Une relation est un sous-ensemble du produit cartésien d'une liste de domaines caractérisé par un nom.

Le modèle relationnel représente l'information dans une collection de relations. Dans le modèle relationnel de données, on optera pour une représentation sous forme de tableau des relations.

2 - Eléments du modèle relationnel

2.1 - Domaine

C'est un ensemble fini de valeurs possibles auquel on définit aussi un ensemble d'opérateurs pouvant être appliqués aux valeurs.

Exemple : Booléen, Entier, Réel, Caractère, Chaîne de caractères, Date, …

2.2 - Attribut

C'est un identificateur (un nom) décrivant une information stockée dans une base de données. En fait, c'est une donnée élémentaire Du système modélisé.

Exemple : Pour la relation Etudiant, on a les attributs Matricule, Nom, Prénom, Date de naissance, Téléphone, Email, …

2.3 - Relation

C'est un sous-ensemble du produit cartésien de n domaines d'attributs (n > 0).

Elle est représentée sous la forme d'un tableau à deux dimensions dans lequel les n attributs correspondent aux titres des n colonnes.

Exemple :

Etudiant	**Matricule**	**Nom**	**Prénom**	**DateNaissance**	**Téléphone**	**Email**
	1	FATI	Bamba	14/09/2000	75 345 91 98	fb@gmail.com
	2	DIAW	Jean	03/12/1998	76 036 78 93	jd@apple.com
	3	SARR	Cheikh	08/09/2002	77 104 57 82	cs@microsoft.com
	4	NDIAYE	Adama	28/02/1980	33 726 93 52	an@amazom.com

2.4 - Uplet ou tuple ou enregistrement

C'est une occurrence d'une relation. On dira aussi qu'elle est une ligne d'une relation.

2.5 - Degré

C'est le nombre d'attributs d'une relation.

2.6 - Cardinalité

C'est le nombre d'uplets ou tuples de la relation.

2.7 - Clé candidate

C'est un ensemble minimal des attributs de la relation dont les valeurs identifient de façon unique une occurrence.

2.8 - Clé primaire

C'est l'une des clés candidates choisie par le concepteur. Elle doit toujours être soulignée.

2.9 - Clé étrangère ou clé secondaire

C'est une clé primaire qui ne se trouve pas dans sa relation d'origine. Elle doit être toujours suivie du symbole #.

3 - Quelques opérations sur les relations

Soient les relations A et B.

3.1 - Union

L'union de A et B noté $\mathbf{A \cup B}$ est l'ensemble des tuples de A et B sans tuples redondantes. Pour que l'union de 2 relations soient possibles, il faut qu'elles aient le même schéma.

3.2 - Intersection

L'intersection de A et B noté $\mathbf{A \cap B}$ est l'ensemble des tuples appartenant à A et à B. Pour que l'intersection de 2 relations soient possibles, il faut qu'elles aient le même schéma.

3.3 - Produit cartésien

Le produit cartésien de A et B noté **AxB** est l'ensemble des tuples de A concaténés à ceux de B. Pour que le produit cartésien de 2 relations soient possibles, il faut qu'elles aient des schémas différents et qu'aucun attribut se soit commun aux 2 relations.

3.4 - Différence

La différence entre A et B noté **A-B** est l'ensemble des tuples qui appartiennent à A et pas à B. Pour que la différence de 2 relations soient possibles, il faut qu'elles aient le même schéma.

Exercice 20 : **Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{42}(100%)**

OBJECTIF 4.3 : PASSER D'UN MODÈLE ENTITÉ-ASSOCIATION À UN MODÈLE LOGIQUE DE DONNÉES RELATIONNEL

Durée : 30 heures	Seul de validation : 80%

Le modèle relationnel de données provient du modèle entité-association. Pour cela, il existe des règles de passage du modèle entité-association au modèle relationnel.

1 - Règles sur les appellations

- Une propriété devient un attribut
- Un identifiant devient une clé
- Un identifiant candidat devient une clé candidate
- Un identifiant primaire devient une clé primaire
- Une entité devient une relation
- Certaines associations deviennent des relations et d'autres disparaissent

2 - Règles de passage des entités à relations

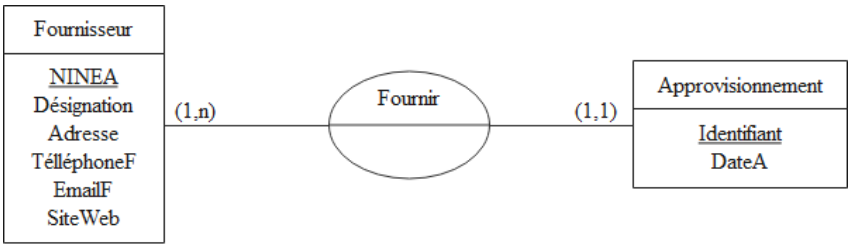
Chaque entité devient une relation avec pour attributs, les propriétés de l'entité et pour clé primaire, l'identifiant primaire de l'entité

3 - Règles de passage des associations à relations

3.1 - Association « père-fils »

L'association disparait et l'identifiant du père migre dans les attributs du fils et devient une clé étrangère.

Exemple



Approvisionnement (**Identifiant**, DateA**, #NINEA**)

Fournisseur (**NINEA**, Désignation, Adresse, TélléphoneF, EmailF, SiteWeb)

Figure 31 : Passage d'une association « père-fils » au modèle relationnel

Dans cette association, Fournisseur est considéré comme me père car sa cardinalité dans la relation est maximale alors que Approvisionnement est considéré comme le fils car sa cardinalité est minimale.

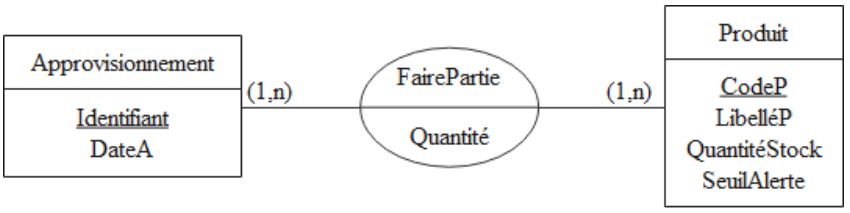
3.2 - Association « plusieurs à plusieurs »

L'association devient une relation avec pour attributs :

- les propriétés de l'association s'il en existait
- les identifiants des différentes entités participant à l'association
- Les identifiants des entités qui migrent dans la relation sont des clés étrangères

  La clé primaire de la relation est la concaténation de toutes les clés étrangères

Exemple



Approvisionnement (Identifiant, DateA)

Produit (CodeP, LibelléP, QuantitéStock, SeuilAlerte)

FairePartie (#Identifiant, #CodeP, Quantité)

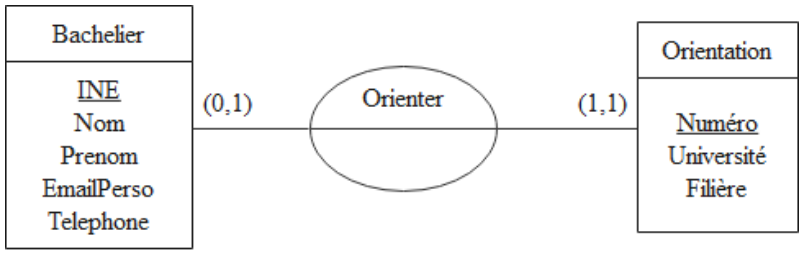
Figure 32 : Passage d'une association « Plusieurs à Plusieurs » au modèle relationnel

3.3 - Association « au plus un » : (0,1) à (1,1)

Ce type d'association est interprétée comme une association « père-fils » où le père est l'entité ayant la cardinalité (0,1) et le fils la cardinalité (1,1)

Exemple

Dans ce système d'orientation des bacheliers dans les universités, on suppose que l'Etat n'oriente pas tous les bacheliers.



Bachelier (**INE**, Nom, Prenom, EmailPerso, Telephone)

Orientation (**Numéro**, Université, Filière, **#INE**)

Figure 33 : Passage d'une association « au plus un : (0,1) à (1,1) » au modèle relationnel

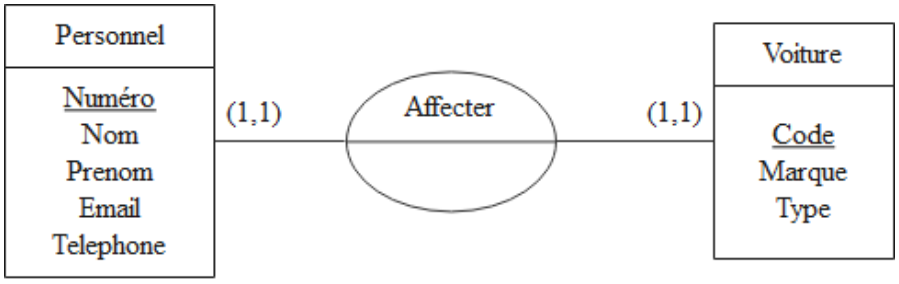
3.4 - Association « un à un » : (1,1) à (1,1) ou « au plus un » : (0,1) à (0,1)

a) Cas d'indépendance existentielle entre les 2 entités

Cette indépendance suppose qu'une donnée de n'importe entité peut exister sans l'autre. Dans ce cas, cette association est interprétée comme une association « plusieurs à plusieurs ».

Exemple

On se place dans un contexte où chaque personnel doit avoir une seule voiture affectée par l'entreprise durant sa mission. L'entreprise commande les voitures indépendamment du personnel et c'est un tirage au sort qui détermine leur affectation.



Personnel (**Numéro**, Nom, Prenom, Email, Telephone)

CoompteUVS (**Code**, Marque, Type)

Affecter (**#Numéro, #Code**)

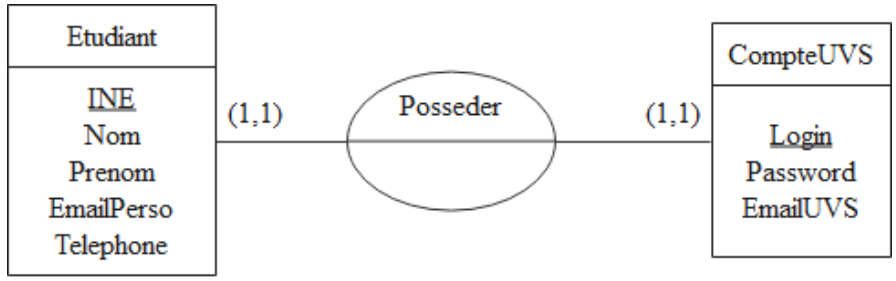
Figure 34 : Passage d'une association « un à un : (1,1) à (1,1) avec indépendance existentielle entre les entités » au modèle relationnel

b) Cas de dépendance existentielle entre les 2 entités

Cette dépendance suppose que la donnée d'une entité ne peut exister avant l'autre. Dans ce cas, cette association est interprétée comme une association « père-fils » où le fils est l'entité dont la donnée dépend de l'autre (le père).

Exemple

Dans le système de l'UVS, il faut être orienté avant la création du compte ; de ce fait le compte ne saurait exister sans étudiant.



Etudiant (**INE**, Nom, Prenom, EmailPerso, Telephone)

CoompteUVS (**Login**, Password, EmailUVS, **#INE**)

Figure 35 : Passage d'une association « un à un : (1,1) à (1,1) avec dépendance existentielle entre les entités » au modèle relationnel

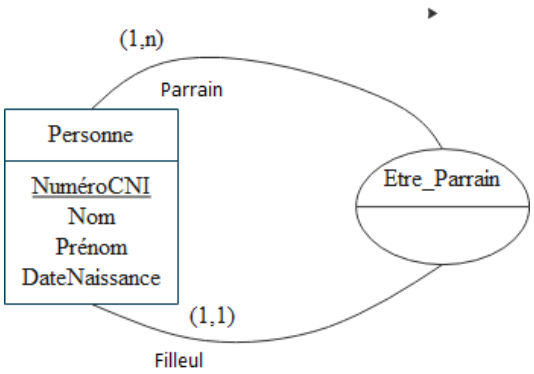
3.5 - Association récursive ou réflexive

En fonction des cardinalités, cette association est interprétée comme une relation « père-fils » ou « plusieurs à plusieurs » ou « au plus un » ou « un à un ».

Les clés (primaires et étrangères) doivent être renommées en utilisant les noms de rôles pour les distinguer.

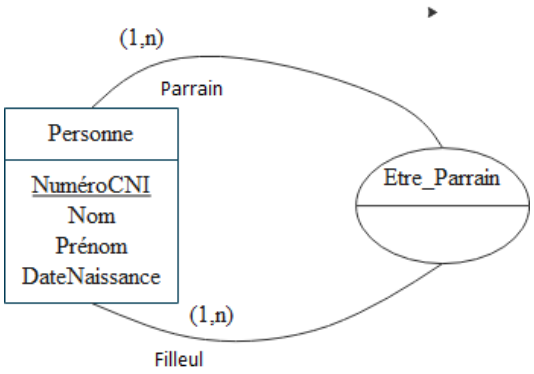
Exemple

Système de parrainage où chaque filleul n'a qu'un seul parrain.



Personne (**NumeroCNI**, Nom, Prénom, DateNaissance, **#NumeroCNIParrain**)

Système de parrainage où un filleul peut avoirs plusieurs parrains.



Personne (**NumeroCNI**, Nom, Prénom, DateNaissance)

Etre_Parrain (#NumeroCNIParrain, #NumeroCNIFilleul)

Figure 36 : Passage d'une association récursive ou réflexive au modèle relationnel

3.6 - Association d'héritage et de spécialisation

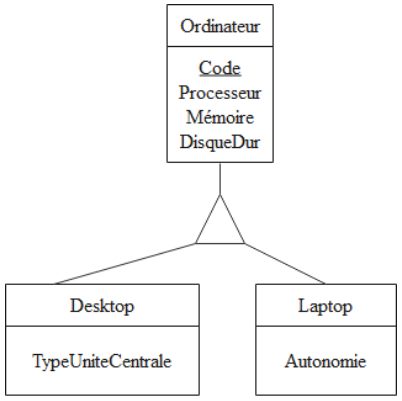
a) Cas d'une spécialisation complète

Une spécialisation est dite complète si toutes les occurrences de l'entité-mère appartiennent à l'instance de l'une des entités-filles.

Dans ce cas, l'entité-mère ne devient pas une relation et chaque entité-fille devient une relation ayant pour attributs, ses propriétés et celle de l'entité-mère et pour clé primaire, l'identifiant de l'entité-mère.

Exemple

Dans cet exemple, la spécialisation est complète car un ordinateur est soit un laptop (ordinateur portable), soit un desktop (ordinateur de bureau)



Desktop (**Code**, Processeur, Mémoire, DisqueDur, TypeUnitéCentrale)

Laptop (**Code**, Processeur, Mémoire, DisqueDur, Autonomie)

Figure 37 : Passage d'une spécialisation complète au modèle relationnel

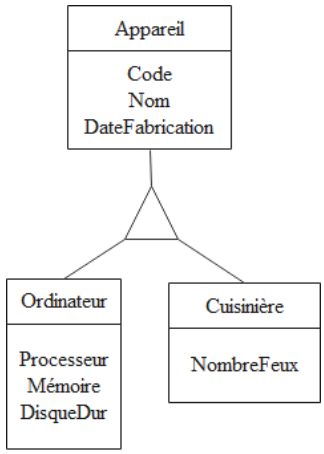
b) Cas d'une spécialisation incomplète

Une spécialisation est dite incomplète si certaines occurrences de l'entité-mère n'appartiennent à l'instance d'aucune des entités-filles

Dans ce cas, l'entité-mère devient une relation et chaque entité-fille devient une relation ayant pour attributs, ses propriétés et celles de l'entité-mère et pour clé primaire, l'identifiant de l'entité-mère.

Exemple

Dans cet exemple, la spécialisation est incomplète car un appareil peut être ni un ordinateur, ni une cuisinière.



Appareil (**Code**, Nom, DateFabrication)

Ordinateur (**Code**, Nom, DateFabrication, Processeur, Mémoire, DisqueDur)

Laptop (**Code**, Nom, DateFabrication, NombreFeux)

Figure 38 : Passage d'une spécialisation complète au modèle relationnel

4 - Contraintes d'intégrité

Ces sont les mêmes contraintes vues précédemment dans le modèle entité-association. A cela, s'ajoute la **contrainte de non nullité de la clé primaire**.

Il existe une valeur conventionnelle **NULL** introduite dans le modèle relationnel pour représenter l'absence d'information. Elle est utilisée pour définir les valeurs des attributs facultatifs dans la relation.

5 - Exemple d'application : e-Commerce

Approvisionnement (**Identifiant**, DateA**, #NINEA**)

Catégorie (**CodeCat**, LibelléCat ; **#CodeCatSup**)

Client (**Reference**, Nom, Prénom, DateNaissance, Sexe, TéléphoneC, EmailC, Login, Password)

Commande (**NuméroCde**, DateCde, **#Reference**)

Contenir (#NuméroCde, #CodeP, QuantitéCde**)**

Facture (**NuméroF**, DateF, **#NuméroCde, #CodeR**)

FairePartie (#Identifiant, #CodeP, Quantité**)**

Fournisseur (**NINEA**, Désignation, Adresse, TéléphoneF, EmailF, SiteWeb)

Livrer (#NuméroF, # CodeP, QuantitéLivrée**)**

ModeReglement (**CodeR**, LibelléR)

Produit (**CodeP**, LibelléP, PrixUnitaire, QuantitéStock, SeuilAlerte, **#CodeCat**)

Figure 39 : Modèle relationnel de données du site d'e-commerce

Exercice 21 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{43}(20%)

Exercice 22 : Seuil de validation : 60% Titre : Modèle Logique de Données Relationnel de la gestion d'un institut de formation Objectifs visés : O_{31}(80%)

Exercice 23 : Seuil de validation : 60% Titre : Modèle Logique de Données Relationnel de la gestion des états civils Objectifs visés : O_{31}(80%)

Exercice 24 : Seuil de validation : 60% Titre : Modèle Logique de Données Relationnel de la gestion d'une chaîne hôtelière Objectifs visés : O_{31}(80%)

OBJECTIF 4.4 : NORMALISER UN SCHÉMA RELATIONNEL

Durée : 30 heures	Seul de validation : 40%

1 - Introduction

La normalisation d'un Modèle Logique de Données Relationnel (ou un schéma de données relationnel ou d'une base de données) est une opération qui consiste à corriger les défauts de conception c'est-à-dire :

- **minimiser les redondances de données** : une donnée peut se retrouver inutilement dans plusieurs relations ;
- **éliminer les incohérences des données** : une occurrence d'une donnée peut être modifiée mais pas les autres ;
- **éliminer les pertes de données** : une information peut être supprimée alors qu'elle est une référence pour les autres ;
- **améliorer la performance des traitements**.

Ces défauts de conception sont dus au non-respect des règles de la modélisation conceptuelle et logique des données. La normalisation garantit la modification des donnés dans une relation (table plus tard) unique et une propagation de ses mises à jour dans les autres relations.

Normaliser une base de données consiste à normaliser chaque relation du MLDR en leur mettant en forme normale de telle enseigne que les dépendances fonctionnelles entre les attributs d'une relation aient pour source que les clés. Il existe 8 formes normales :

1. Première Forme Normale : 1FN
2. Deuxième Forme Normale : 2FN
3. Troisième Forme Normale : 3FN
4. Forme Normale de Boyce-Codd : FNBC
5. Quatrième Forme Normale : 4FN
6. Cinquième Forme Normale : 5FN
7. Forme Normale Domaine Clé : FNDC
8. Sixième Forme Normale :6FN

dont les plus utilisées sont : 1FN, 2FN ; 3FN et FNBC.

Ainsi une base normalisée est une base de données qui respecte les formes normales les plus utilisées.

La forme normale d'une base de données est sa forme normale la plus élevée.

2 - Première Forme Normale :1FN

Une relation est en Première Forme Normale si tous ses attributs sont atomiques c'est-à-dire élémentaire non décomposable. Il faut se rappeler que toute relation possède une clé primaire.

Exemple : Toutes les relations de la Figure 39 sont en 1FN

Contre-exemple : Soit la relation suivante

Personne(Matricule, Nom, Adresse, DateNaissance)

Figure 40 : Exemple de relation qui n'est en 1FN

Elle n'est pas en 1FN car les attributs Nom et Adresse ne sont pas atomiques car d'une part, les occurrences de l'attribut Nom contiennent le prénom et le nom de famille et d'autre part, les occurrences de l'attribut Adresse contiennent la boîte postale, la ville et le pays (BP 15 126 Dakar, Sénégal)

Pour rendre une relation en 1FN, il faut décomposer tous les attributs non atomiques en attributs atomiques.

Ainsi, la relation Personne sera :

Personne(Matricule, Prenom, NomFamille, BoitePostale, Ville, Pays, DateNaissance)

Figure 41 : Normalisation en 1FN de la Figure 40

Attention : L'atomicité d'une valeur dépend de son contexte d'utilisation.

Par exemple, s'il n'est pas nécessaire de décomposer une adresse en boîte postale, ville et pays dans son contexte d'utilisation alors l'attribut Adresse est atomique

3 - Deuxième Forme Normale :2FN

Une relation en Deuxième Forme Normale si :

- elle est en 1FN ;
- toute dépendance fonctionnelle entre la clé et les autres attributs n'appartenant pas à la clé sont élémentaires c'est-à-dire tout attribut n'appartenant pas à la clé dépend fonctionnellement de toute la clé et non d'une partie seulement.

Ainsi, une relation en 1FN et ayant une clé formée d'un seul attribut est donc en 2FN.

Exemple : Toutes les relations de la Figure 39 sont en 2FN

R(A, B, C) avec A,B→C et B→C

Figure 42 : Contre-exemple typique d'une relation qui n'est pas en 2FN

Contre-exemple : Soit la relation représentant l'évaluation des étudiants :

Examen(INE, CodeExamen, Date, Nom, Prenom, NomMatiere, SalleExamen)

Figure 43 : Exemple de relation qui n'est pas en 2FN

Cette relation en 2FN car SalleExamen, Nom et Prénom dépendent fonctionnellement d'une partie de la clé : CodeExamen → SalleExamen et INE → Nom, Prenom

Pour rendre une relation en 2FN, il faut :

- **mettre la relation en 1FN ;**
- **décomposer la relation en plusieurs relations de telle manière à regrouper dans une relation tous les attributs non clés qui ne dépendent pas fonctionnellement de la clé entière**
- **ajouter à ces nouvelles relations comme clé, la partie de la clé dont dépend les attributs de la nouvelle relation**
- **ajouter dans la relation initiale comme clé étrangère, les clés des nouvelles relations**

La normalisation en 2FN de la Figure 43 donne les relations suivantes :

Examen(#INE, #CodeExamen, Date, NomMatiere)

Etudiant(INE, Nom, Prenom)

Salle(CodeExamen, SalleExamen)

Figure 44 : Normalisation en 2FN de la Figure 43

Le respect de la 2FN évite une redondance des données qui encombrent inutilement la mémoire et l'espace disque.

4 - Troisième Forme Normale :3FN

Une relation en Troisième Forme Normale si :

- elle est en 2FN ;
- tout attribut n'appartenant pas à une clé ne dépend pas d'un autre attribut non clé c'est-à-dire tout attribut non-clé dépend directement d'une clé.

Attention : Il s'agit de toute clé candidate pas seulement de la clé primaire

Exemple : Toutes les relations de la Figure 39 sont en 3FN

R(A, B, C) avec A→B et B→C

Figure 45 : Contre-exemple typique d'une relation qui n'est pas en 3FN

Contre-exemple : Soit la relation représentant la fabrication des produits :

Fabrication(CodeProduit, NomProduit, DateFabrication, CodeUsine, NomUsine)

Figure 46 : Exemple de relation qui n'est pas en 3FN

Cette relation n'est pas en 3FN car l'attribut CodeUsine n'est ni clé primaire, ni clé candidate alors que CodeUsine → NomUsine.

Pour rendre une relation en 3FN, il faut :

- **mettre la relation en 2FN ;**
- **décomposer la relation en plusieurs relations de telle manière à regrouper dans une relation tous les attributs non clés qui ne dépendent pas fonctionnellement de la clé (clé candidate)**
- **ajouter à ces nouvelles relations comme clé, l'attribut dont dépend les attributs de la nouvelle relation**
- **ajouter dans la relation initiale comme clé étrangère, les clés des nouvelles relations**

La normalisation en 3FN de la Figure 46 donne les relations suivantes :

Fabrication(CodeProduit, NomProduit, DateFabrication, #CodeUsine)

Usine(CodeUsine, NomUsine)

Figure 47 : Normalisation en 3FN de la Figure 43

5 - Forme Normale de Boyce-Codd : FNBC ou 3,5FN

Une relation en Forme Normale de Boyce-Codd si :

- elle est en 3FN ;
- tout attribut qui n'appartient pas à une clé (candidate) n'est pas source d'une dépendance fonctionnelle vers une partie d'une clé. C'est-à-dire que les seules dépendances fonctionnelles élémentaires existantes sont celles dans lesquelles une clé détermine un attribut.

Exemple : Toutes les relations de la Figure 39 sont en FNBC

R(A, B, C) avec A,B→C et C→B

Figure 48 : Contre-exemple typique d'une relation qui n'est pas en FNBC

Contre-exemple : Soit la relation suivante :

Transport(Itinéraire, Immatriculation, Date, Chauffeur, Rapport)

Avec les dépendances fonctionnelles suivantes :
NumeroItineraire, Immatriculation, Date → CodeChauffeur, Rapport
CodeChauffeur → Immatriculation

Figure 49 : Exemple de relation qui n'est en FNBC

Cette relation n'est pas en FNBC car une voiture est affectée à un seul chauffeur (CodeChauffeur → Immatriculation)

Pour rendre une relation en FNBC, il faut :

- **mettre la relation en 3FN ;**
- **décomposer la relation en plusieurs relations de telle manière à regrouper dans une relation tous les attributs figurant dans la dépendance fonctionnelle dont un attribut non-clé détermine une partie de la clé ;**
- **mettre comme clé dans les relations récemment crées ; l'attribut source de la dépendance fonctionnelle ;**
- **supprimer dans la relation initiale, les attributs figurant dans les relations récemment créées ;**
- **mettre comme clé étrangère dans la relation initiale, les clés primaires des relations récemment crées.**

La normalisation en FNBC de la Figure 49 donne les relations suivantes :

Affectation(CodeChauffeur, Immatriculation)

Transport(Itinéraire, Date, #Chauffeur, Rapport)

Figure 50 : Normalisation en FNBC de la Figure 49

Exercice 25 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{44}(20%)

SÉQUENCE 5 : INTERROGATION DES BASES DE DONNÉES : L'ALGÈBRE RELATIONNELLE

OBJECTIF 5.1 : ECRIRE DES REQUÊTES D'INTERROGATION DE LA BASE DE DONNÉES EN ALGÈBRE RELATIONNELLE

Durée : 72 heures	Seul de validation : 50%

L'algèbre relationnelle est un langage de requêtes dans des bases de données relationnelles. L'algèbre relationnelle a été inventée en 1970 par Edgar Frank Codd, le directeur de recherche du centre IBM de San José.

L'algèbre relationnelle fournit les opérations permettant d'interroger les bases de données. C'est un support mathématique cohérent sur lequel repose le modèle relationnel.

En algèbre relationnelle, on dit comment obtenir un résultat. Ce dernier est généralement une nouvelle relation, on dit ainsi que les opérations de l'algèbre relationnelle sont des opérations **ensemblistes**.

1 - Les opérateurs unaires

Les opérateurs unaires sont les opérateurs qui utilisent une seule relation. En plus de la relation, il est possible d'utiliser d'autres choses comme la condition ou l'attribut. Le terme unaire est utilisé pour signifier le nombre de relations entrant en jeu dans l'opération.

1.1 - La sélection ou restriction

La sélection ou la restriction permet d'extraire d'une relation R, les données qui respectent une condition P. La condition est une expression booléenne dont l'évaluation donnera soit la valeur VRAI soit FAUX. La condition utilise les opérateurs relationnels ou de comparaison suivants : l'égalité (=), la différence ou l'inégalité (≠), l'infériorité ou sens stricte (<), l'infériorité au sens large (≤), la supériorité au sens stricte (>) et la supériorité au sens large (≥). La condition peut être simple ou composée en utilisant les opérateurs logiques ou booléennes suivantes : la conjonction logique (**ET**), la disjonction logique (**OU**) et la négation logique (**NON**).

L'opérateur de la sélection est σ et est défini ainsi :

$$\sigma : \text{Relation x Condition} \rightarrow \text{Relation}$$
$$(R, P) \longmapsto R'$$

Et s'écrit ainsi : $R' = \sigma_P(R)$ ou $R' = \sigma_{[P]}(R)$

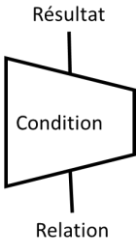


Figure 51 : Représentation graphique de l'opérateur de sélection ou restriction

R' a le même schéma de données que R mais l'instance de R' est une partie de l'instance de R dont les données vérifient la condition P.

Exemple : Sélection des produits dont qu'il faut approvisionner le stock dans l'urgence en se basant sur le schéma de données de la Figure 39 : Modèle relationnel de données du site d'e-commerce

$$\boldsymbol{Resultat} = \sigma_{\mathbf{QuantiteStock <= SeuilAlerte}}(\mathbf{Produit})$$

Figure 52 : Représentation graphique de la requête de sélection des produits à approvisionner en urgence

1.2 - La projection

La projection permet de créer une nouvelle relation dont les attributs correspondent à une partie des attributs de la relation initiale. Les données de la nouvelle relation sont les données de la relation initiale dont les attributs sont désignés dans la nouvelle relation.

L'opérateur de la projection est π et est défini ainsi :

$$\begin{array}{|c|}\hline \pi : \mathbf{Relation\ x\ Liste\ d'attributs} \rightarrow \mathbf{Relation} \\ (\mathbf{R}, A_1, A_2, \dots, A_n) \longmapsto \mathbf{R'} \\ \hline \end{array}$$

Et s'écrit ainsi : $\boldsymbol{R'} = \pi_{A_1,A_2,\dots,A_n}(\mathbf{R})$ $\boldsymbol{ou}$ $\boldsymbol{R'} = \pi_{[A_1,A_2,\dots,A_n]}(\mathbf{R})$

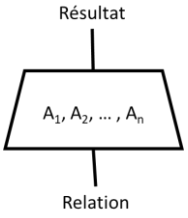


Figure 53 : Représentation graphique de l'opérateur de projection

Exemple : Liste des clients avec les attributs : nom, prénom, téléphone et adresse électronique en se basant sur le schéma de données de la Figure 39 : Modèle relationnel de données du site d'e-commerce

$$\boldsymbol{Résultat} = \boldsymbol{\pi}_{\mathbf{Nom,\ Prenom,\ TelephoneC,\ Email}}(\mathbf{Client})$$

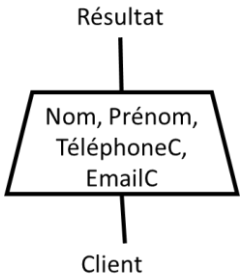


Figure 54 : Représentation graphique de la requête de la projection de la relation Client sur les attributs : nom, prénom, téléphone et adresse électronique

1.3 - Le renommage

Le renommage permet de changer le nom de certains attributs d'une relation sans modifier la structure de la relation. La nouvelle relation obtenue après renommage de certains attributs a des données identiques à celles de la relation initiale.

L'opérateur du renommage est α et est défini ainsi :

$$\boldsymbol{\alpha} : \mathbf{Relation\ x\ Liste\ d'attributs\ avec\ le\ nouveauu\ nom} \rightarrow \mathbf{Relation}$$
$$(\mathbf{R}, \boldsymbol{A_1}: \mathbf{NouveauNom_1},\ \boldsymbol{A_2}: \boldsymbol{NouveauNom_2}, \dots, \boldsymbol{A_n}: \boldsymbol{NouveauNom_n}) \longmapsto \mathbf{R'}$$

Et s'écrit ainsi : $\boldsymbol{R'} = \boldsymbol{\alpha}_{\boldsymbol{A_1}:\mathbf{NouveauNom_1},\ \boldsymbol{A_2}:\boldsymbol{NouveauNom_2},\dots,\boldsymbol{A_n}:\boldsymbol{NouveauNom_n}}(\mathbf{R})$ **ou**

$$\boldsymbol{R'} = \boldsymbol{\alpha}_{[\boldsymbol{A_1}:\mathbf{NouveauNom_1},\ \boldsymbol{A_2}:\boldsymbol{NouveauNom_2},\dots,\boldsymbol{A_n}:\boldsymbol{NouveauNom_n}]}(\mathbf{R})$$

Exemple : Liste des fournisseurs où la désignation est renommée par nom du fournisseur en se basant sur le schéma de données de la Figure 39 : Modèle relationnel de données du site d'e-commerce

$$\boldsymbol{R'} = \boldsymbol{\alpha}_{\mathbf{Designation\ :Nom\ du\ fournisseur}}(\mathbf{Fournisseur})$$

2 - Les opérateurs binaires

Les opérateurs binaires sont les opérateurs qui utilisent deux relations. En plus des relations, il est possible d'utiliser d'autres choses comme la condition ou l'attribut. Le terme binaire est utilisé pour signifier le nombre de relations entrant en jeu dans l'opération.

2.1 - L'union

L'union de 2 relations de même schéma est une relation de même schéma dont les données sont les données des 2 relations. L'union est une opération commutative et associative.

L'opérateur d'union est ∪ et est défini ainsi :

$$\cup : \textbf{Relation x Relation} \rightarrow \textbf{Relation}$$
$$(\mathbf{R_1}, \mathbf{R_2}) \longmapsto \mathbf{R'}$$

Et s'écrit ainsi : $\mathbf{R' = R_1 \cup R_2}$

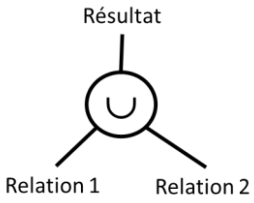


Figure 55 : Représentation graphique de l'opérateur d'union

2.2 - L'intersection

L'intersection de 2 relations de même schéma est une relation de même schéma dont les données sont celles identiques aux 2 relations. L'intersection est une opération commutative et associative.

L'opérateur d'intersection est ∩ et est défini ainsi :

$$\cap : \textbf{Relation x Relation} \rightarrow \textbf{Relation}$$
$$(\mathbf{R_1}, \mathbf{R_2}) \longmapsto \mathbf{R'}$$

Et s'écrit ainsi : $\mathbf{R' = R_1 \cap R_2}$

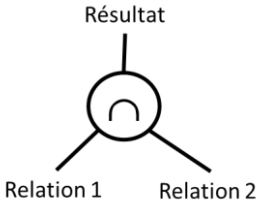


Figure 56 : Représentation graphique de l'opérateur d'intersection

2.3 - La différence

La différence de 2 relations R_1 et R_2 de même schéma est une relation de même schéma dont les données sont celles de la première relation (R_1) qui ne sont pas dans la seconde (R_2). La différence n'est ni commutative, ni associative.

L'opérateur de la différence est - et est défini ainsi :

$$\begin{array}{|c|} \hline - : \textbf{Relation x Relation} \rightarrow \textbf{Relation} \\ (\mathbf{R_1}, \mathbf{R_2}) \mapsto \mathbf{R'} \\ \hline \end{array}$$

Et s'écrit ainsi : $\mathbf{R' = R_1 - R_2}$

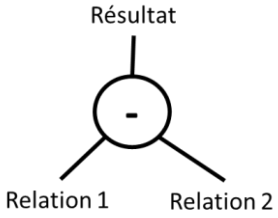


Figure 57 : Représentation graphique de l'opérateur de différence

2.4 - Le produit cartésien

Le produit cartésien de 2 relations de schémas différents avec une intersection vide des 2 schémas est une relation dont le schéma est la concaténation des schémas des 2 relations et les données sont obtenues en concaténant chaque tuple de la première relation avec chaque tuple de la seconde relation. Le produit cartésien est une opération commutative et associative.

L'opérateur du produit cartésien est **X** et est défini ainsi :

$$\begin{array}{|c|} \hline \boldsymbol{X} : \textbf{Relation x Relation} \rightarrow \textbf{Relation} \\ (\mathbf{R_1}, \mathbf{R_2}) \mapsto \mathbf{R'} \\ \hline \end{array}$$

Et s'écrit ainsi : $\mathbf{R' = R_1\ X\ R_2}$

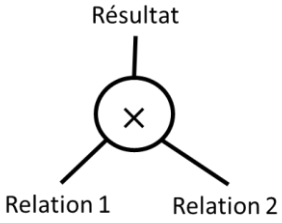


Figure 58 : Représentation graphique de l'opérateur du produit cartésien

2.5 - La division

La division de la relation R_1 par la relation R_2 est une relation dont les attributs sont ceux de R_1 qui ne sont pas dans R_2 et les données sont des valeurs correspondant aux attributs du résultat tels qu'en faisant le produit cartésien avec les données de R_2, on obtient les tuples de inclus dans R_1. Pour que la division soit possible, il faut que les attributs de R_2 soient inclus dans les attributs de R_1 et que R_1 aient plus d'attributs que R_2. La division n'est ni commutative, ni associative.

L'opérateur de la division est ÷ et est défini ainsi :

$$\begin{array}{|c|} \hline \div : \textbf{Relation x Relation} \quad \rightarrow \textbf{Relation} \\ (\mathbf{R_1}, \mathbf{R_2}) \qquad\qquad \longmapsto \mathbf{R'} \\ \hline \end{array}$$

Et s'écrit ainsi : $\mathbf{R' = R_1 \div R_2}$

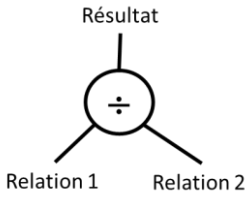


Figure 59 : Représentation graphique de l'opérateur de la division

3 - Les opérateurs composés : les jointures

3.1 - La jointure naturelle

La jointure naturelle de 2 relations est une relation dont le schéma est la concaténation des 2 schémas sans répétition des attributs identiques et les données sont obtenues en concaténant chaque tuple de la première relation avec chaque tuple de la seconde relation ayant les mêmes valeurs sur les attributs communs c'est-à-dire que tous les attributs communs aux 2 relations doivent avoir les mêmes valeurs. La jointure naturelle est une opération commutative et associative.

La jointure naturelle est généralement utilisée pour 2 relations ayant une contrainte d'intégrité référentielle entre elles c'est-à-dire que l'une des 2 relations comporte une clé étrangère venant de l'autre relation. La jointure naturelle se fera selon la condition que la clé étrangère soit égale à la clé primaire de l'autre relation.

L'opérateur de la jointure naturelle est ⋈ et est défini ainsi :

$$\begin{array}{|c|} \hline \bowtie : \textbf{Relation x Relation} \quad \rightarrow \textbf{Relation} \\ (\mathbf{R_1}, \mathbf{R_2}) \qquad\qquad \longmapsto \mathbf{R'} \\ \hline \end{array}$$

Et s'écrit ainsi : $\mathbf{R' = R_1 \bowtie R_2}$

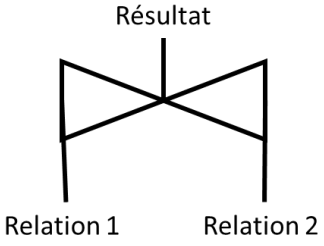


Figure 60 : Représentation graphique de l'opérateur de la jointure naturelle

La jointure naturelle peut être obtenue par :

- un produit cartésien des 2 relations. Pour faire le produit cartésien, il faut renommer l'un des attributs identiques.
- une restriction sur les attributs communs (sachant que certains attributs ont été renommés)
- une projection pour éliminer les attributs renommés

3.2 - La théta-jointure ou inéqui-jointure

La théta-jointure naturelle de 2 relations est une relation dont le schéma est la concaténation des 2 schémas et les données sont obtenues en concaténant chaque tuple de la première relation avec chaque tuple de la seconde relation vérifiant la condition P. Dans la condition P, les opérateurs de comparaison possibles ne peuvent être que $<$, $\leq$, $>$, $\geq$, et $\neq$. La théta-jointure est une opération commutative et associative.

L'opérateur de la théta-jointure est $\bowtie_\theta$ et est défini ainsi :

$$\bowtie_\theta : \mathbf{Relation\ x\ Relation\ x\ Condition} \rightarrow \mathbf{Relation}$$
$$(\mathbf{R_1}, \mathbf{R_2}, P) \longmapsto \mathbf{R'}$$

Et s'écrit ainsi : $\mathbf{R'} = \mathbf{R_1} \bowtie_{\theta[P]} \mathbf{R_2}$

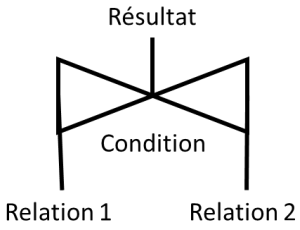


Figure 61 : Représentation graphique de l'opérateur de la théta-jointure ou inéqui-jointure

La théta-jointure peut être obtenue en calculant le produit cartésien des 2 relations suivi d'une restriction vérifiant la condition.

3.3 - L'équi-jointure

L'équi-jointure est une théta-jointure ou le seul opérateur de comparaison est l'égalité (=)

Exercice 26 : **Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{51}(20%)**

Exercice 27 : **Seuil de validation : 60% Titre : Requêtes d'interrogation de la gestion d'un institut de formation Objectifs visés : O_{51}(80%)**

Exercice 28 : **Seuil de validation : 60% Titre : Requêtes d'interrogation de la gestion des états civils Objectifs visés : O_{51}(80%)**

Exercice 29 : **Seuil de validation : 60% Titre : Requêtes d'interrogation de la gestion d'une chaîne hôtelière Objectifs visés : O_{51}(80%)**

SÉQUENCE 6 : IMPLÉMENTATION D'UNE BASE DE DONNÉES : MODÈLE PHYSIQUE

OBJECTIF 6.1 : DÉCRIRE LES PRINCIPES DE LA GESTION INFORMATISÉE D'UNE BASE DE DONNÉES

Durée : 06 heures	Seul de validation : 60%

Le modèle relationnel des données produit un schéma de données relationnel qui faut écrire dans un langage compréhensible par les SGBD relationnelles. Le langage le plus utilisé est **SQL** : **Structured Query Language** (**Langage de requêtes structurées**) qui sert à :

- implémenter la base de données
- sécuriser la base de données
- manipuler les données
- réaliser un ensemble de requêtes en une transaction

1 - Le langage SQL (Structured Query Language)

SQL est un langage d'interrogation structurée de données conçu dans les années 70. Il est devenu au travers de ses différentes versions, un langage complet de gestion de bases de données relationnelles. Pour gérer la base de données, SQL dispose de 4 catégories de langages.

1.1 - Le Langage de Définition des Données (LDD) : Data Definition Language (DDL)

Il comporte l'ensemble des commandes SQL de gestion de la structure de chaque objet (table, vue, …) de la base de données notamment :

- la création (CREATE) des objets
- la modification (ALTER) des objets
- la suppression (DROP) des objets
- le renommage (RENAME) des objets
- …

1.2 - Le Langage de Manipulation des Données (LMD) : Data Manipulation Language (DML)

Il comporte l'ensemble des commandes SQL réalisant les opérations suivantes sur les données :

- l'ajout (INSERT) des données dans une table
- la mise à jour (UPDATE) des données d'une table
- la suppression (DELETE) des données d'une table
- la visualisation (SELECT) des données d'une ou plusieurs tables

1.3 - Le Langage de Contrôle des Données (LCD) : Data Control Language (DCL)

Il comporte l'ensemble des commandes SQL de gestion des droits d'accès aux tables et vues :

- l'autorisation (GRANT) d'accès
- l'interdiction (REVOKE) d'accès

1.4 - Le Langage de Contrôle des Transactions (LCT) : Transaction Control Language (TCL)

Une transaction est un ensemble de requêtes SQL indivisible qui met à jour une base de données en garantissant sa cohérence. Elle comporte un début, une suite de requêtes SQL et une fin. Les transactions doivent respecter les principes résumés par l'acronyme **ACID** :

a) Atomicité

Une transaction doit être complètement exécutée ou totalement annulée. Dans ce sens, une transaction constitue une unité d'exécution dont les opérations doivent être annulées lorsque la transaction est interrompue. L'atomicité obéit au principe du « tout ou rien »

b) Cohérence

Il est possible qu'une transaction en cours d'exécution transgresse temporairement certaines contraintes de cohérence. Mais quand la transaction est terminée, le résultat final doit respecter de nouveau toutes les contraintes. Par conséquent, une transaction fait passer une base de données d'un état cohérent à un autre état cohérent et garantit l'absence de données contradictoires. La transaction représente donc une unité d'exécution qui doit maintenir la cohérence d'une base de données. La cohérence signifie l'absence de données contradictoires.

c) Isolation

Le principe de l'isolation exige que les modifications apportées à la base de données par les opérations d'une transaction en cours ne doivent être visibles qu'après leur validation. Ainsi, une donnée en cours de modification ne doit pas être accessible. L'isolation protège la base de données contre les effets de bord.

d) Durabilité

Une base de données doit être maintenue dans un état cohérent jusqu'à la validation des modifications effectuées par une transaction. Lors d'incidents (erreurs de programmation, interruptions du système ou pannes d'unités de stockage externes), la durabilité garantit que seule une transaction correctement exécutée et validée permet de valider de façon durable les modifications faites sur la base de données par les différentes opérations de la transaction.

Le principe ACID, fondamental aux SGBD garantit que chaque utilisateur transforme une base de données d'un état cohérent à un autre état cohérent. Les états intermédiaires incohérents restent invisibles depuis l'extérieur (à la transaction) et sont annulés en cas d'incident.

Une transaction n'a que deux issues possibles : le succès (COMMIT) ou l'échec (ROLLBACK). Il n'y a pas de situation intermédiaire.

Seuls les requêtes SQL qui modifient des données interviennent dans une transaction. Une requête SELECT qui n'effectue qu'une lecture n'influe pas sur la cohérence des données.

2 - Les différentes étapes d'implémentation d'une base de données

L'implémentation de la base de données est la transcription du modèle relationnel de données dans un SGBD pour obtenir un modèle physique de données. Pour y arriver, il y a des étapes à respecter.

2.1 - Choix du Système de Gestion de Bases de Données Relationnelles (SGBDR)

Il faut choisir un SGBD relationnel car le modèle à transcrire est de type relationnel. Il en existe plusieurs :

- **SGBD propriétaire (payant)** : Microsoft Access, Oracle Database, Microsoft SQL Server, DB2, MaxDB, 4D, dBase, Informix, Sybase
- **SGBD libre ("gratuit")** : MySQL, PostgreSQL, MariaDB, Open Office Base, Firebird, Ingres, HSQLDB, Derby, Apache Derby, …

2.2 - La création de la base de données

Elle se fait en utilisant le Langage de Définition des Données pour transcrire les différentes relations du modèle relationnel.

Certains SGBD apportent des spécificités au langage SQL. Si vous utilisez le SQL standard alors vos requêtes seront exécutables sur toutes les SGBD.

2.3 - La sécurisation de la base de données

Il est question de créer des utilisateurs de la base de données et leur attribuer des droits d'accès aux tables et vues. Elle se fait en utilisant le Langage de Contrôle de Données.

3 - Exploitation d'une base de données

Une fois que le schéma de la base de données est mis sur place, les utilisateurs doivent y insérer des données, les mettre à jour, les visualiser voire les supprimer. Cette exploitation se fait en utilisant le Langage de Manipulation de Données et le Langage de Contrôle de Transactions.

Exercice 30 : **Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{61}(100%)**

OBJECTIF 6.2 : IMPLÉMENTER UNE BASE DE DONNÉES

Durée : 48 heures	Seul de validation : 75%

Dans le cadre de ce cours, nous utiliserons le SGBD Microsoft Access ou Open Office Base ou MySQL.

1 - Téléchargement et installation du SGBD

Je vous recommande d'utiliser Microsoft Access mais vous pouvez utiliser les autres SGBD.

1.1 - Installation de Microsoft Access

https://www.youtube.com/watch?v=VkLUyZMIjC8

1.2 - Installation de Open Office Base

https://www.youtube.com/watch?v=BLzAKwlz2KQ

1.3 - Installation de MySQL

https://www.youtube.com/watch?v=K1SOagDC3Xg

2 - Création de la base de données

Pour des raisons de volume horaire affecté à ce cours, nous n'écrirons pas les requêtes (Langage de Définition des Données) de création de la base de données. Nous utiliserons les interfaces graphiques des SGBD à cet effet. Un prochain cours insistera sur le Langage de Définition de Données.

La vidéo suivante vous présentera la création de la base de données, ses tables et ses contraintes d'intégrité référentielle avec le SGBD Microsoft Access.

https://www.youtube.com/watch?v=WAqDtyx-Ny4

3 - Exemple d'application : e-Commerce

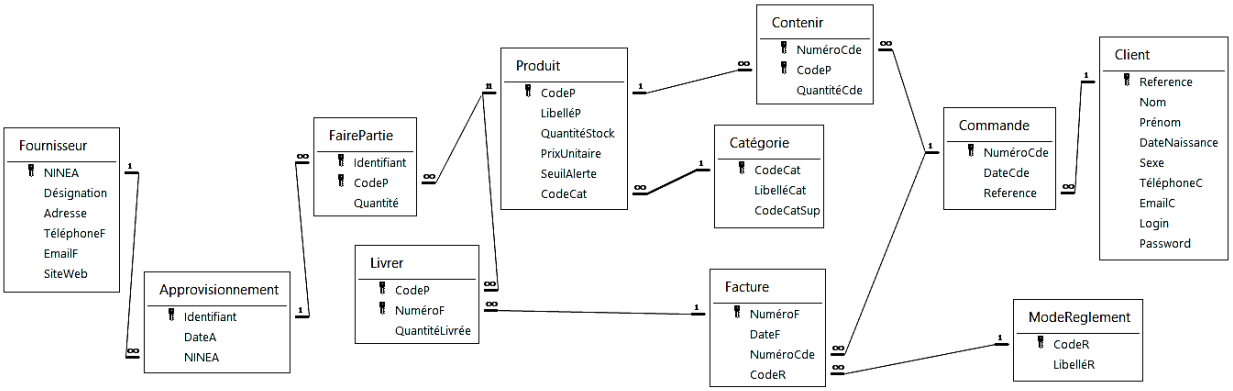


Figure 62 : Représentation du Modèle Physique de Données du site e-Commerce réalisé avec le logiciel Microsoft Access

4 - Insertion des données dans la base de données

L'insertion des données se fait en utilisant la commande **INSERT** du Langage de Manipulation des Données (LMD) dont la syntaxe est la suivante :

INSERT INTO nom_table(colonne1, colonne2, …) **VALUES**(valeur1, valeur2, …) ;

Figure 63 : Syntaxe SQL pour l'insertion des données dans une table

La liste des colonnes et les parenthèses en rouge peuvent être omises à condition que toutes les valeurs des colonnes soient citées et qu'elles respectent l'ordre des colonnes dans la table.

Exemple : Insertion des données dans la table Fournisseur

INSERT INTO Fournisseur (NINEA, Désignation, Adresse, TéléphoneF, EmailF, SiteWeb) **VALUES** (456321, "Ets Bon Marché", "Kafountine", "+221 33 694 12 00", "bon.marche@commerce.sn", "http://www.bonmarche.sn") ;

INSERT INTO Fournisseur **VALUES** (345680, "Ets Meilleur Choix", "Lac Rose", "+221 33 994 56 37", "meilleur.choix@commerce.sn", " http://www.meilleurchoix.sn") ;

Figure 64 : Exemples d'insertion des données dans la table Fournisseur

Il est possible d'insérer plusieurs lignes ou tuples de données à la fois dans une table. Les lignes sont séparées par des virgules.

Exemple : Insertion de plusieurs tuples dans la table Fournisseur

```
INSERT INTO Fournisseur (NINEA, Désignation, Adresse, TéléphoneF, EmailF, SiteWeb) VALUES
(456321, "Ets Bon Marché", "Kafountine", "+221 33 694 12 00", "bon.marche@commerce.sn", "http://www.bonmarche.sn"),
(783245, "Quincaillerie des experts", "Sindone", "+221 33 012 56 37", "venez@commerce.sn", "https://www.avous.sn")
(345680, "Ets Meilleur Choix", "Lac Rose", "+221 33 994 56 37", "meilleur.choix@commerce.sn", "http://www.meilleurchoix.sn") ;
```

Figure 65 : Exemple d'insertion de plusieurs lignes ou tuples dans la table Fournisseur

Fournisseur					
NINEA	**Désignation**	**Adresse**	**TéléphoneF**	**EmailF**	**SiteWeb**
345680	Ets Meilleur Choix	Lac Rose	+221 33 994 56 37	meilleur.choix@commerce.sn	http://www.meilleurchoix.sn
456321	Ets Bon Marché	Kafountine	+221 33 694 12 00	bon.marche@commerce.sn	http://www.bonmarche.sn
783245	Quincaillerie des experts	Sindone	+221 33 012 56 37	venez@commerce.sn	https://www.avous.sn

Figure 66 : Résultat de l'insertion multiple dans la table Fournisseur

5 - Modification des données dans la base de données

La modification des données se fait en utilisant la commande **UPDATE** du Langage de Manipulation des Données (LMD) dont la syntaxe est la suivante :

```
UPDATE nom_table SET Colonne1=Valeur1, …, ColonneN=ValeurN WHERE condition ;
```

Figure 67 : Syntaxe SQL pour la modification des données d'une table

Dans la modification, il faut préciser les colonnes à modifier et les valeurs.

La clause WHERE et la condition sont optionnelles et dans ce cas, toutes les données de la table seront modifiées.

Exemple : Modification de données de la table Fournisseur

```
UPDATE Fournisseur SET Adresse="Rosso" WHERE NINEA=456321 ;
```

Figure 68 : Exemple de modification des données de table Fournisseur

Fournisseur					
NINEA	**Désignation**	**Adresse**	**TéléphoneF**	**EmailF**	**SiteWeb**
345680	Ets Meilleur Choix	Lac Rose	+221 33 994 56 37	meilleur.choix@commerce.sn	http://www.meilleurchoix.sn
456321	**Ets Bon Marché**	**Rosso**	**+221 33 694 12 00**	**bon.marche@commerce.sn**	**http://www.bonmarche.sn**
783245	Quincaillerie des experts	Sindone	+221 33 012 56 37	venez@commerce.sn	https://www.avous.sn

Figure 69 : La table Fournisseur après modification de l'adresse du fournisseur de NINEA 456321

Dans la requête ci-dessus, il est question de modifier par Rosso toutes les lignes dont le NINEA est 456321.

6 - Suppression des données dans la base de données

La suppression des données se fait en utilisant la commande **DELETE** du Langage de Manipulation des Données (LMD) dont la syntaxe est la suivante :

```
DELETE FROM nom_table WHERE condition ;
```

Figure 70 : Syntaxe SQL pour la suppression des données d'une table

La clause WHERE et la condition sont optionnelles et dans ce cas, toutes les données de la table seront supprimées.

Exemple : Suppression de données de la table Fournisseur

```
DELETE FROM Fournisseur WHERE NINEA=456321 ;
```

Figure 71 : Exemple de suppression des données de table Fournisseur

Fournisseur					
NINEA	**Désignation**	**Adresse**	**TéléphoneF**	**EmailF**	**SiteWeb**
345680	Ets Meilleur Choix	Lac Rose	+221 33 994 56 37	meilleur.choix@commerce.sn	http://www.meilleurchoix.sn
783245	Quincaillerie des experts	Sindone	+221 33 012 56 37	venez@commerce.sn	https://www.avous.sn

Figure 72 : La table Fournisseur après suppression du fournisseur de NINEA 456321

Exercice 31 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{62}(25%)

Exercice 32 : Seuil de validation : 60% Titre : Modèle Physique de Données de la gestion d'un institut de formation Objectifs visés : O_{52}(20%)

Exercice 33 : Seuil de validation : 60% Titre : Modèle Physique de Données de la gestion des états civils Objectifs visés : O_{62}(20%)

Exercice 34 : Seuil de validation : 60% Titre : Modèle Physique de Données de la gestion d'une chaîne hôtelière Objectifs visés : O_{62}(20%)

Exercice 35 : Seuil de validation : 60% Titre : Manipulation des données de la gestion d'un institut de formation Objectifs visés : O_{62}(60%)

Exercice 36 : Seuil de validation : 60% Titre : Manipulation des données de la gestion des états civils Objectifs visés : O_{62}(60%)

Exercice 37 : Seuil de validation : 60% Titre : Manipulation des données de la gestion d'une chaîne hôtelière Objectifs visés : O_{62}(60%)

OBJECTIF 6.3 : INTERROGER UNE BASE DE DONNÉES EN LANGAGE SQL

Durée : 78 heures	Seul de validation : 75%

L'interrogation d'une base de données consiste à extraire des données respectant un ensemble de conditions. La commande permettant d'interroger la base de données est **SELECT** et à la syntaxe générale suivante :

SELECT ensemble de colonnes séparées par une virgule
FROM ensemble de tables
WHERE conditions sur les colonnes
GROUP BY ensemble de colonnes
HAVING conditions sur les colonnes figurant dans la clause GROUP BY
ORDER BY ensemble de colonnes séparées par une virgule **;**

- **SELECT** permet de lister les colonnes à afficher. Il est possible de mettre seulement une étoile (*****) et dans ce cas, toutes les colonnes de toutes les tables concernées par la requête seront affichées.
- **FROM** permet de lister les tables concernées par la requête. Il faudra tenir compte des contraintes d'intégrité référentielles (liaisons ou jointures) entre les tables
- **WHERE** permet de lister des conditions que doivent respecter les données à extraire
- **GROUP BY** permet de faire des regroupements de données en fonction des colonnes précises. Le regroupement est utilisé lorsqu'on utilise dans le SELECT des fonctions d'agrégation comme la somme (sum), la moyenne (avg), le comptage (count), etc.
- **HAVING** permet de lister des conditions que doivent respecter les données regroupées (figurant dans le GROUP BY)
- **ORDER BY** permet d'ordonner les données en fonction des colonnes par ordre croissant (ASC) ou décroissant (DESC)

Remarque : Une requête SQL se termine toujours par un point-virgule (**;**).

Dans WHERE et HAVING, les conditions sont exprimées avec les opérateurs suivants :

- **Opérateurs arithmétiques** : +, -, *, /
- **Opérateurs logiques** : NOT, AND, OR
- **Opérateurs de comparaison** : =, <>, <, <=, >, >=, IN, BETWEEN, LIKE, EXISTS

Il existe en langage SQL, un ensemble d'opérateurs issus de l'algèbre relationnelle permettant interrogation d'une base de données.

Avant de continuer, il faut remplir les différentes tables de la base de données et y tester les requêtes qui seront données en exemple.

1 - La sélection ou restriction

La sélection ou restriction permet d'extraire dans une ou plusieurs tables des données qui respectent une ou plusieurs conditions.

Exemple 1 : Sélection des clients

```
SELECT * FROM Client ;
```

Exemple 2 : Sélection des produits dont qu'il faut approvisionner le stock dans l'urgence

```
SELECT * FROM Produit WHERE QuantiteStock <= SeuilAlerte ;
```

Dans le résultat d'une requête, il peut exister un tuple (ligne ou enregistrement) en plusieurs occurrences. Ainsi, juste après le SELECT, il est possible de mettre :

- **DISTINCT** : pour afficher une seule occurrence de chaque tuple
- **ALL** : pour afficher toutes les occurrences de chaque tuple

Exemple 3 : Sélection de toutes livraisons des produits facturés

```
SELECT ALL * FROM Livrer ;
```

Exemple 4 : Sélection distincte (unique)de toutes livraisons des produits facturés

```
SELECT DISTINCT * FROM Livrer ;
```

2 - La projection

La projection permet de préciser les colonnes à afficher dans les résultats de la requête.

Exemple 1 : Sélection des noms et prénoms des clients

```
SELECT DISTINCT Nom, Prenom FROM Client ;
```

Exemple 2 : Sélection des noms et prénoms des clients de sexe féminin

```
SELECT DISTINCT Nom, Prenom FROM Client WHERE Sexe="F" ;
```

3 - Le renommage

Le renommage permet de renommer les colonnes dans une requête. On le fait généralement pour des colonnes obtenues par calcul ou pour des colonnes dont le nom n'est pas assez expressif. Pour renommer une colonne, il faut la suivre du mot-cle **AS** puis du nouveau nom. Si le nouveau nom contient de l'espace, il faudra le mettre entre côtes

Exemple 1 : Sélection des produits et préciser la valeur marchande de chaque produit (Quantité en stock X Prix unitaire)

```
SELECT *, QuantiteStock * PrixUnitaire AS "Valeur Marchande" FROM Produit ;
```

Exemple 2 : Sélection des factures de janvier 2021

```
SELECT NumeroF AS Numéro, DateF AS "Date Facture", NuméroCde AS "Numéro Commande", CodeR AS "Code Règlement"
FROM Facture
WHERE DateF BETWEEN 01/01/20201 AND 31/01/2021 ;
```

4 - L'union

L'union permet de fusionner (combiner) les tuples de 2 requêtes à condition qu'elles aient le même schéma (les mêmes colonnes dans le même ordre). Le résultat aura le même schéma que les 2 requêtes et les éventuels doublons seront supprimés.

Exemple : Liste des téléphones et emails des clients et des fournisseurs

```
(SELECT TelephoneC AS Telephone, EmailC AS Email FROM Client)
UNION
(SELECT TelephoneF AS Telephone, EmailF AS Email FROM Fournisseur) ;
```

5 - L'intersection

L'intersection permet de sélectionner des tuples identiques à 2 requêtes à condition qu'elles aient le même schéma. Le résultat aura le même schéma que les 2 requêtes

Exemple : Liste des téléphones et emails de ceux qui sont clients et fournisseurs à la fois

```
(SELECT TelephoneC AS Telephone, EmailC AS Email FROM Client)
INTERSECT
(SELECT TelephoneF AS Telephone, EmailF AS Email FROM Fournisseur) ;
```

Attention : INTERSECT n'est pas implémenté dans les SGBD MySQL et Microsoft Access. Dans ce cas, il faut utiliser l'opérateur **EXISTS** ou **IN**.

6 - La différence

La différence permet de sélectionner les tuples de la première requête qui ne sont pas dans la seconde à condition que les 2 requêtes aient le même schéma. Le résultat aura le même schéma que les 2 requêtes

Exemple : Liste des téléphones et emails des clients qui ne sont pas des fournisseurs

```
(SELECT TelephoneC AS Telephone, EmailC AS Email FROM Client)
MINUS
(SELECT TelephoneF AS Telephone, EmailF AS Email FROM Fournisseur) ;
```

Attention : MINUS n'est pas implémenté dans le SGBD Microsoft Access. Dans ce cas, il faut utiliser l'opérateur **NOT EXISTS** ou **NOT IN**.

7 - Le produit cartésien

Le produit cartésien permet d'obtenir tous les tuples tels que chaque tuple de la première table est concaténé à chaque tuple de la seconde table. Le schéma du résultat sera le schéma de la première table concaténé au schéma de la seconde table.

Exemple : Liste des catégories et des modes de règlement concaténés

```
SELECT * FROM Catégorie, ModeReglement ;
```

8 - La jointure

La jointure est une opération binaire qui s'appuie sur le produit cartésien de 2 tables ou vues pour extraire des données correspondantes à un ensemble de conditions. Le schéma du résultat sera le schéma de la première table concaténée au schéma de la seconde table. En fonction des conditions, on a plusieurs types de jointure.

8.1 - La jointure interne : INNER JOIN

La jointure interne est une opération qui permet d'extraire les tuples ou enregistrements qui satisfont une condition dans les 2 tables.

a) L'équi-jointure

L'équi-jointure est une jointure sur 2 tables avec une condition d'égalité. Elle permet d'extraire les tuples des 2 tables qui satisfont la condition d'égalité (pas forcément sur les clés primaire et étrangère).

Exemple 1 : Equi-jointure entre les tables Produit et Contenir

```
SELECT * FROM Produit INNER JOIN Contenir ON Produit.CodeP = Contenir.CodeP ;
```

L'qui-jointure peut aussi s'écrire sans le terme « INNER JOIN ». En fait, on fait un produit cartésien des 2 tables puis une restriction sur l'égalité de 2 colonnes.

```
SELECT * FROM Produit, Contenir WHERE Produit.CodeP = Contenir.CodeP ;
```

L'exemple 1 n'a aucun sens puisque nous ne faisons pas la théorie des bases de données. En fait, il ne vous sera presque jamais demandé de faire l'équi-jointure de 2 tables mais vous

devez interpréter la question pour comprendre qu'il faut une équijointure ou autre. L'exemple 2 est plus concret et proche de la réalité.

Exemple 2 : Liste des produits commandés

```
SELECT * FROM (Produit INNER JOIN Contenir ON Produit.CodeP = Contenir.CodeP)
INNER JOIN Commande ON Commande.NuméroCde = Contenir.NuméroCde ;
```

Le « SELECT * » n'est pas correct car il est demandé la liste des produits et non la liste des Produit, Contenir et Commande. Il faut donc faire une projection sur les colonnes de la table Produit.

```
SELECT Produit.* FROM (Produit INNER JOIN Contenir ON Produit.CodeP = Contenir.CodeP)
INNER JOIN Commande ON Commande.NuméroCde = Contenir.NuméroCde ;
```

L'exemple 2 serait plus réaliste si on devait extraire les produits appartenant à une commande particulière.

Exemple 3 : Liste des produits (Code et libellé) appartenant à la commande numéro A48

Dans cet exemple, on fera une projection sur les colonnes CodeP et LibelleP puis faire une restriction sur la colonne NuméroCde = "A48"

```
SELECT Produit.CodeP, LibelléP
FROM (Produit INNER JOIN Contenir ON Produit.CodeP = Contenir.CodeP)
 INNER JOIN Commande ON Commande.NuméroCde = Contenir.NuméroCde
WHERE Commande.NuméroCde = "A48" ;
```

b) La théta-jointure ou la non-équijointure ou l'inéqui-jointure

La théta-jointure est une jointure sur 2 tables avec une condition d'inégalité (>, >=, <, <=, <>, IN, LIKE, BETWEEN … AND …, EXISTS). Elle permet d'extraire les tuples des 2 tables qui satisfont la condition d'inégalité.

Exemple 1 : Liste des clients dont leur nom est contenu dans la désignation du fournisseur

```
SELECT * FROM Cleint INNER JOIN Fournisseur ON Désignation LIKE "*" & Nom & "*" ;
```

La théta-jointure n'est pas supportée par le SGBD Microsoft Access. Il faudra faire un produit cartésien puis une restriction.

c) La jointure naturelle : NATURAL JOIN

La jointure naturelle entre 2 tables est possible s'il y a au moins une colonne qui porte le même nom entre les 2 tables. La jointure naturelle est une équi-jointure entre les 2 tables

avec des conditions d'égalité sur les différentes colonnes identiques des 2 tables. Elle permet de faire une jointure sans préciser les colonnes concernées.

Exemple : Jointure naturelle entre les tables Produit et Contenir

```
SELECT * FROM Produit NATURAL JOIN Contenir ;
```

La jointure naturelle n'est pas supportée par le SGBD Microsoft Access

8.2 - La jointure externe : OUTER JOIN

La jointure externe est une opération qui permet d'extraire les tuples ou enregistrements d'une part qui satisfont la condition dans les 2 tables et d'autre part certains tuples qui ne satisfont pas la condition dans les 2 tables.

a) La jointure externe gauche : LEFT JOIN ou LEFT OUTER JOIN

La jointure externe gauche permet d'extraire tous les enregistrements de la table de gauche (LEFT) même si la condition n'est pas vérifiée dans l'autre table tout en mettant à NULL les valeurs des colonnes de la table de droite.

Exemple : Liste de tous les Produits avec les quantités commandées

```
SELECT Produit.*, QuantitéCde
FROM Produit LEFT JOIN Contenir ON Produit.CodeP = Contenir.CodeP ;
```

b) La jointure externe droite : RIGHT JOIN ou RIGHT OUTER JOIN

La jointure externe droite permet d'extraire tous les enregistrements de la table de droite (RIGHT) même si la condition n'est pas vérifiée dans l'autre table tout en mettant à NULL les valeurs des colonnes de la table de gauche.

Exemple : Liste des commandes avec les clients y compris ceux qui n'ont pas commandés

```
SELECT *
FROM Commande RIGHT JOIN Client ON Commande.Reference = Client.Reference ;
```

La jointure externe droite peut être obtenue à partie de la jointure externe gauche en permutant juste l'ordre des 2 tables.

```
SELECT *
FROM Client LEFT JOIN Commande ON Commande.Reference = Client.Reference ;
```

c) La jointure externe complète : FULL JOIN ou FULL OUTER JOIN

La jointure externe complète permet d'extraire les enregistrements quand la condition est vraie dans au moins une des 2 tables tout en mettant à NULL les valeurs des colonnes de table qui ne respecte pas la condition.

Exemple : Jointure externe complète entre les tables Commande et Client

```
SELECT *
FROM Commande FULL JOIN Client ON Commande.Reference = Client.Reference  ;
```

La jointure externe complète peut être obtenue en faisant l'union du résultat de la jointure externe gauche avec le résultat de la jointure externe droite.

La jointure externe complète n'est pas supportée par le SGBD Microsoft Access.

8.3 - La jointure croisée : CROSS JOIN

La jointure croisée permet de faire le produit cartésien de 2 tables.

Exemple : Liste des Produits concaténés aux Clients

```
SELECT * FROM Produit CROSS JOIN Client ;
```

8.4 - L'auto-jointure

L'auto-jointure permet d'effectuer une jointure (interne, externe ou croisée) d'une table avec elle-même comme si c'était une autre table. Pour y parvenir, il faut créer des alias de la table

Exemple : Liste des sous-catégories de la catégorie Informatique

```
SELECT SousCat.CodeCAt, SousCat.LibelléCat
FROM Catégorie SupCat
 INNER JOIN Catégorie SousCat ON SupCat.CodeCatSup = SousCat.CodeCat ;
```

SupCat est un alias de la table Catégorie qui représente les catégories supérieures.

SousCat est également un alias de la table Catégorie qui représente les sous-catégories.

L'alias d'une table se créé juste en mettant le nom de l'alias devant cette table. Si vous le désirez, vous pouvez créer des alias de table dans toutes vos requêtes ; son avantage est d'avoir des noms plus courts ou simplifiés.

Exercice 38 : Seuil de validation : 80% Titre : Test de connaissance Objectifs visés : O_{63}(20%)

Exercice 39 : Seuil de validation : 60% Titre : Interrogation des données de la gestion d'un institut de formation Objectifs visés : O_{63}(60%)

Exercice 40 : **Seuil de validation : 60% Titre : Interrogation des données de la gestion des états civils Objectifs visés : O_{63}(60%)**

Exercice 41 : **Seuil de validation : 60% Titre : Interrogation des données de la gestion d'une chaîne hôtelière Objectifs visés : O_{63}(60%)**