REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université TAHRI Mohammed Béchar
Faculté des Sciences Exactes
Département des Mathématique et Informatique

N° d'ordre : UTMB/FSE/P.....

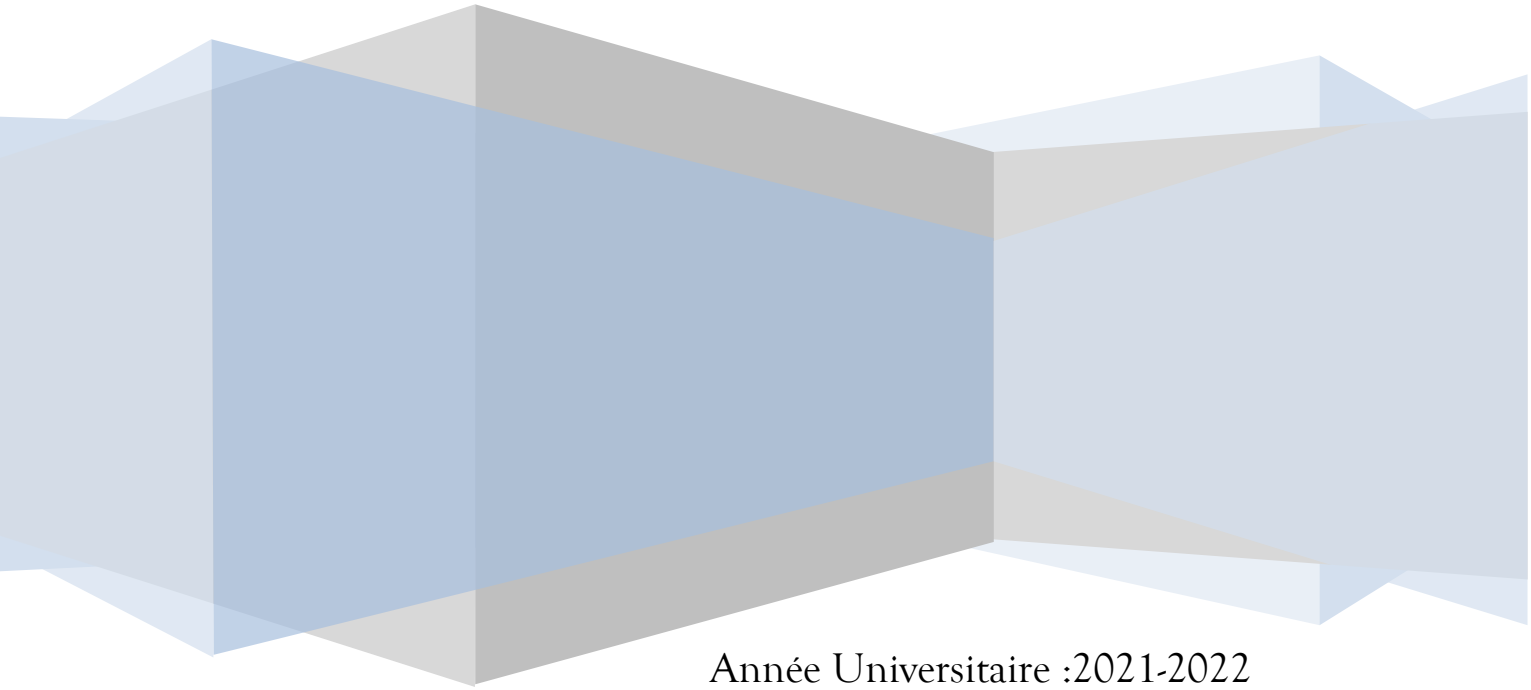
Filière : Informatique

Spécialité : Systèmes Informatiques et Réseaux.

Module : Bases de Données Avancées

Bases de Données Avancées

Dr. Seddiki Noureddine



Année Universitaire :2021-2022

SOMMAIRE

Liste des figures

Liste des tableaux

Résumé

Ce polycopié est un support de cours de base de données avancée ; destiné principalement aux étudiants de première année master en informatique de la spécialité MSIR. Il peut aussi servir aux étudiants de MSIA et MIAD.

Ce polycopié contient les notions fondamentales qu'un étudiant de première année master en informatique doit absolument connaitre.

Ce polycopié est composé de cinq chapitres :

- La première partie, est consacrée à un rappel sur les bases de données relationnelles.
- La deuxième partie énumère quelques notions sur les bases de données orientée objets.
- La troisième partie traite les bases de données reparties.
- La quatrième partie nous présentons les bases de données multimédia.
- La cinquième partie, discute les bases de données déductives.

Chapitre I
Rappel sur Le modèle relationnel

I. Rappel sur Le modèle relationnel

1. Introduction

Le modèle relationnel a été formalisé par entreprise CODD en 1970. Quelques exemples de réalisation il sont : DB2(IBM), INFORMIX, INGRES, ORACLE.

Dans ce modèle, les données sont stockées dans des tables, sans préjuger de la façon dont les informations sont stockées dans la machine. Un ensemble de données sera donc modélisé par un ensemble de tables.

Le succès du modèle relationnel auprès des chercheurs, concepteurs et utilisateurs est dû à la puissance et à la simplicité de ses concepts. En outre, contrairement à certains autres modèles, il repose sur des bases théoriques solides, notamment la théorie des ensembles et la logique mathématique (théorie des prédicats d'ordre 1).

Les objectifs du modèle relationnel :

- Proposer des schémas de données faciles à utiliser,
- Améliorer l'indépendance logique et physique,
- Mettre à la disposition des utilisateurs des langages de haut niveau pouvant éventuellement être utilisés par des non informaticiens,
- Optimiser les accès à la base de données,
- Améliorer l'intégrité et la confidentialité,
- Fournir une approche méthodologique dans la construction des schémas.

2. Les modèles de données

2.1. Définition

Nous avons défini un modèle de données comme un ensemble de concepts et de règles de composition de ces concepts permettant de décrire des données. Il existe trois modèles de données :

- le modèle hiérarchique;
- le modèle réseau;
- le modèle relationnel;

Historiquement les deux premiers modèles ont été conçus dans la période 1965-1970, et ont donné naissance à plusieurs systèmes commercialisés : IMS (Information Management System) utilisant le modèle hiérarchique, IDS(Integrated Data Store) utilisant le modèle réseau…etc.

L'organisation des données au sein d'une BD a une importance essentielle pour faciliter l'accès et la mise à jour des données.

2.2. Le modèle réseau

Le modèle réseau a été proposé par le groupe DBTG du comité CODASYL dans la période 1965-1970, Il a été amélioré à plusieurs reprises grâce à des études qui ont abouti surtout à sa normalisation.

Les éléments de base de ce modèle sont:

- Les ensembles d'entités (Enregistrement logique);
- L'association entre les ensembles d'entités (liens).

Enregistrement logique et lien

La notion d'enregistrement logique n'est pas spécifique aux SGBD, c'est celle que l'on rencontre dans la plupart des systèmes de gestion de fichiers et des langages de programmation. Elle permet de regrouper des données élémentaires en enregistrements logiques.

Exemple1 : Personne = enregistrement

NSS : entier, clé

Nom : caractère(10)

Prénom : caractère(10)

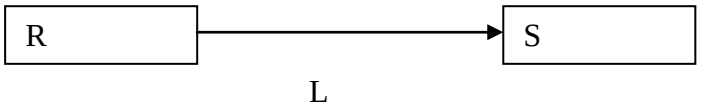
Adresse : caractère(40)

Fin

Le deuxième concept fondamental dans le modèle réseau est le lien. Un lien peut se définir comme la représentation d'une association, l'association est une perception abstraite de la réalité alors que le lien va être sa matérialisation.

Un lien (relation) L est défini entre deux types d'enregistrements R et S dans une direction donnée. Il offre un mécanisme d'accès qui, à partir d'un enregistrement r de type R, permet d'accéder aux enregistrements s1, s2, …sn de type S qui lui sont reliés. Conventionnellement, un lien est

représenté par un diagramme appelé diagramme de bachman, du nom de l'inventeur, où sont mentionnés les types d'enregistrement R et S et le lien L qui va de R vers S.



FIGI.1. lien entre deux enregistrements

Exemple2:

Soit le diagramme suivant (**FIGI.2**) qui associé les entités 'Enseignant' et 'Département ' par le lien 'Est attacher à
,

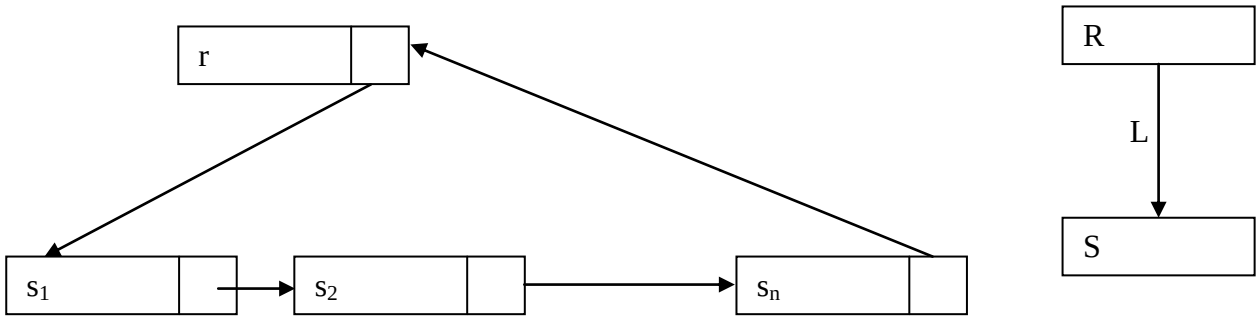
FIGI.2. Exemple d'entités et de lien

A. Le modèle réseau CODASYL

Le lien appelé 'SET' dans la terminologie CODASYL, ne peut être utilisé que lorsque l'association qui existe entre les enregistrements de type R et S fait correspondre à un enregistrement r les enregistrements s1, s2.... ,sn alors qu'inversement à chaque élément sI ne correspond au plus qu'un élément r. Dans ce cas, on pourra construire un lien directionnel de R vers S.

Cela veut dire, qu'un lien CODASYL pourra être construit si l'association est de type 1-1, 1-n ou n-1. Dans le cas d'une association de type m-n, il faudrait introduire un enregistrement de nouveau type (voir ci-dessous) ; avant cela, expliquons comment se fait le mécanisme d'accès.

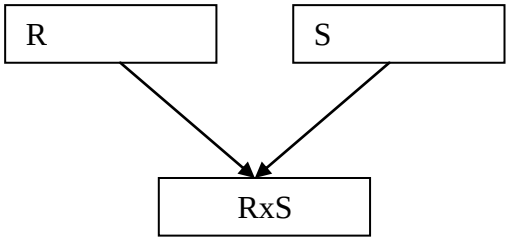
Le mécanisme d'accès qui permet de passer de l'enregistrement r aux enregistrements s1, s2, ….,sn est celui d'une liste circulaire (voir **FIGI.2**) où la tête de liste est constituée par l'enregistrement r et les éléments de la liste sont constitués par les enregistrements s1, s2, ….,sn



FIGI.3. Représentation d'un lien CODASYL à l'aide d'une liste circulaire

Cette représentation est possible car chaque enregistrement de type S n'étant associé qu'à un seul enregistrement de type R, et ne peut apparaître que dans une seule liste circulaire. En effet, si un enregistrement de type S était parcouru par plusieurs listes, il y aurait un nombre variable de pointeurs et c'est la raison pour laquelle une association de type m-n ne peut être représentée par un lien CODASYL sans une transformation préalable.

Dans le cas d'une association de type m-n entre deux types d'enregistrements R et S, la solution consiste à se ramener à deux associations de type 1-n en construisant un enregistrement de type noté RxS (**FIGI.3**) dont les éléments sont tous les couples de la forme (rI,sj) si rI est de type R, sj de type S et rI et sj étant reliés entre eux dans l'association. Si on examine maintenant les associations qui existent respectivement entre RxS et R et S, on remarque qu'elles sont toutes deux de type 1-n



FIGI.4. Représentation de l'association m-n en formalisme CODASYL

B. Schémas réseaux

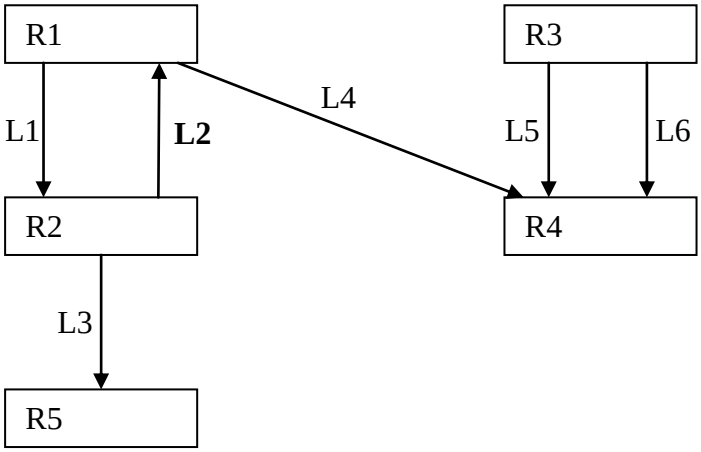
La notion de lien CODASYL telle qu'elle vient d'être définie va permettre de construire des schémas réseaux (**FIGI.4**) dont la structure sera spécifique de chaque application, mais qui d'une façon générale auront les propriétés suivantes :

a. d'un enregistrement de type R peuvent partir autant de liens différents que l'on veut.

b. à un enregistrement de type S peuvent arriver autant de liens que l'on veut.

c. entre deux enregistrements de type R et S, il peut y avoir plusieurs liens différents allant de R vers S, et inversement.

d. pour un enregistrement de type R, un lien partant de R ne peut boucler sur R. Cette limitation a en fait été levée dans certains systèmes commercialisés.



FIGI.5. Exemple de diagramme réseau CODASYL

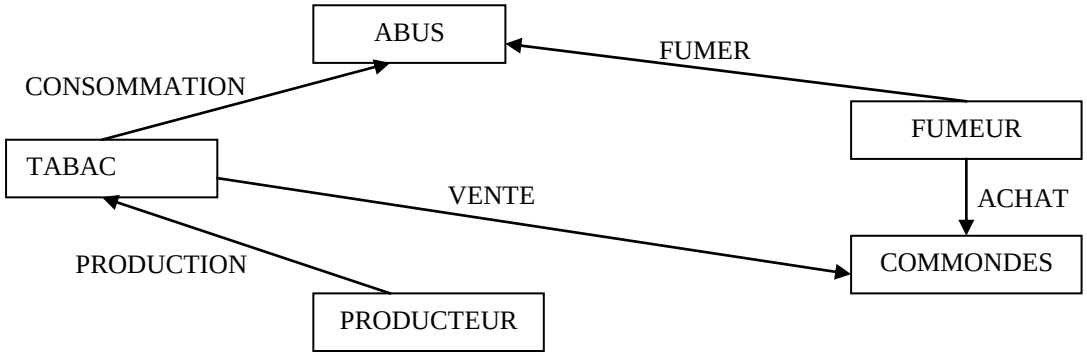
A titre d'exemple, nous allons présenter le diagramme CODASYL d'une base de données 'toxicommane' (fig.II.6) composée des enregistrements suivants :

- TABAC composé des données élémentaires suivantes : Numéro-Tabac et Nom-Tabac
- FUMEUR composés des données : Numéro-Fumeur, Nom et Prénom
- ABUS décrivant pour chaque tabac, la quantité fumée par un fumeur
- PRODUCTEUR définissant pour chaque tabac, le nom du producteur
- COMMANDES spécifiant les commandes de tabac passées par les fumeurs aux producteurs

Les liens existants entre ces différents enregistrements sont :

- PRODUCTION qui associe un producteur aux tabacs produits
- VENTE qui associe un tabac aux commandes correspondantes
- ACHAT qui associe un fumeur à ses commandes
- FUMER qui associe un fumeur à une quantité de tabac fumé

- CONSOMMATION qui associe un tabac à toutes ses quantités fumées par les différents fumeurs.



FIGI.6. Diagramme de la BDD réseau 'toxicomane'

C. Les opérations de manipulation dans un modèle réseau

Il y a deux grandes catégories d'opérations :

1. Des opérations de de modification :

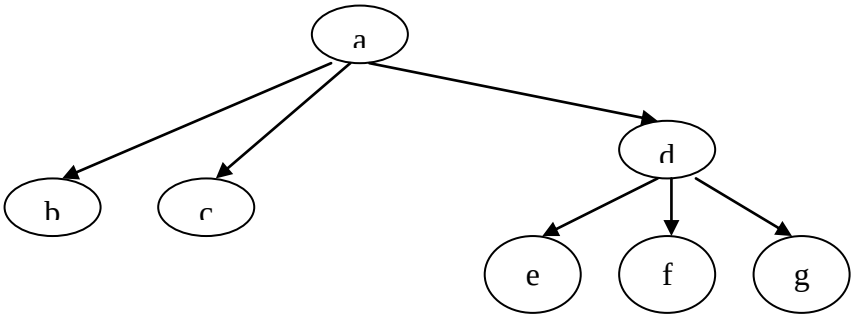
   Sont des opérations qui permettent de modifier le contenu de la BDD tell que la création, la suppression et de mise à jour

2. Et autre qui permet de sélectionner de l'information

2.3. Le modèle Hiérarchique

Les concepts de base de ce modèle sont les enregistrements logiques qui sont reliés entre eux pour constituer une arborescence. Une arborescence est un cas particulier d'un graphe réseau dans lequel un nœud, peut recevoir au plus une flèche, et d'où il peut partir un nombre quelconque de flèche, comme le montre la **FIGI.6.**

Sur l'arborescence, il existe un nœud particulier : la racine, c'est un nœud qui ne reçoit aucune flèche. On appelle fils d'un nœud l'ensemble des nœuds qui se trouvent à une distance 1 de ce nœud. On appelle père d'un nœud, la notion inverse de celle de fils. Dans une arborescence, tout nœud n'a qu'un seul père et peut avoir plusieurs fils.



FIGI.7. Exemple d'un modèle hiérarchique (arborescence)

3. Les concepts fondamentaux du modèle

A. Notion de Domaine

C'est un ensemble de valeurs caractérisées par un nom.

Exemple3:

NOM_VILLE = { Alger, Oran, Béchar }

B. Notion de relation

Sous-ensemble du produit cartésien d'une liste de domaines caractérisé par un nom.

En d'autres termes, une relation n'est ni plus ni moins qu'une table dans laquelle chaque colonne correspond à un domaine et porte un nom ce qui rend leur ordre sans aucune importance

Par exemple, à partir des domaines :

D1={Dép_Inf, Dép_Ang, Dép_Chi} et D2={Informaticien, Anglais, Chimiste}

La relation associer à de deux domaine :

Ensg_Dép	**D1**	**D2**
	Dép_Inf	Informaticien
	Dép_Ang	Anglais
	Dép_Chi	Chimiste

- Une relation peut être vue comme un tableau à deux dimensions dont les colonnes correspondent aux domaines et les lignes contiennent les tuples. On parle parfois de table relationnelle.

C. Notion de tuple (n-uplet)

Est une ligne d'une relation

Exemple: (Dép_Inf, Informaticien) c'est une ligne de la relation **Ensg_Dép** qui correspond un tuple

D. Notion d'attribut

Colonne d'une relation caractérisée par un nom.

Exemple 3: N°SS, Nom, Prénom, Age, Adresse

E. Notion de schéma de relation

Nom de la relation, suivi de la liste des attributs avec leurs domaines.

F. Notion de Système de gestion de bases de données relationnel

C'est un logiciel supportant le modèle relationnel, et qui peut manipuler les données avec des opérateurs relationnels.

Exemple 4:

Une relation décrivant des personnes et comportant les attributs : N°SS, Nom, Prénom, Age, Adresse : PERSONNE (N°SS, Nom, Prénom, Age, Adresse).

G. Notion de base de données relationnelle

Base de données dont le schéma est un ensemble de schémas de relations et dont les occurrences sont les tuples de ces relations.

Ou une base de données est alors un ensemble de relations (extensions) associé au schéma de base de données et vérifiant ses contraintes d'intégrité.

4. Les dépendances fonctionnelles(DF)

A. Notion de dépendance fonctionnelle

Dans une relation quelconque, un attribut Y dépend fonctionnellement d'un attribut X, si pour une valeur de X, il ne peut exister, qu'une valeur de Y.

Y dépend fonctionnellement de X, ou X détermine Y.

Définition

Soit R une relation qui contint les attributs X1, X2, ….Xn un schéma de relation et A et B des sous-ensembles de { X1, X2, ….Xn}. On dit que A détermine B ou B dépend fonctionnellement de A, si étant donné une valeur de A, il lui correspond une valeur unique de B.

Exemple5:
Soit la relation voiture, on a les dépendances fonctionnelles :
N°V $\rightarrow$ couleur
type $\rightarrow$ marque
type$\rightarrow$ puissance

(type,marque) → puissance

B. Propriétés des dépendances fonctionnelles

Dans la notion des dépendances fonctionnelles il existe plusieurs règles :

1. **Réflexivité :** $Y \subseteq X \Rightarrow X \rightarrow Y$; cette règle stipule que tout ensemble d'attributs détermine lui-même ou une partie de lui-même.

2. **Augmentation :** $X \rightarrow Y \Rightarrow XZ \rightarrow YZ$; cette règle signifie que si X détermine Y, les deux ensembles d'attributs peuvent être enrichis par un troisième.

3. **Transitivité :** $X \rightarrow Y$ et $Y \rightarrow Z \Rightarrow X \rightarrow Z$

Les trois règles précédentes composent les axiomes des DF. Plusieurs autres règles se déduisent des axiomes :

4. **Union :** $X \rightarrow Y$ et $X \rightarrow Z \Rightarrow X \rightarrow YZ$

5. **Pseudo-transitivité :** $X \rightarrow Y$ et $WY \rightarrow Z \Rightarrow WX \rightarrow Z$

6. **Décomposition :** $X \rightarrow Y$ et $Z \subseteq Y \Rightarrow X \rightarrow Z$

A partir de ces règles, il est possible d'introduire la notion de dépendance fonctionnelle élémentaire :

C. Notion de Dépendance Fonctionnelle Elémentaire

C'est une DF de la forme $X \rightarrow A$, où A est un attribut unique non inclus dans X ($A \not\subset X$) et où il n'existe pas $X' \subset X$ tel que $X' \rightarrow A$

La seule règle d'inférence qui s'applique aux dépendances fonctionnelles élémentaires est la transitivité.

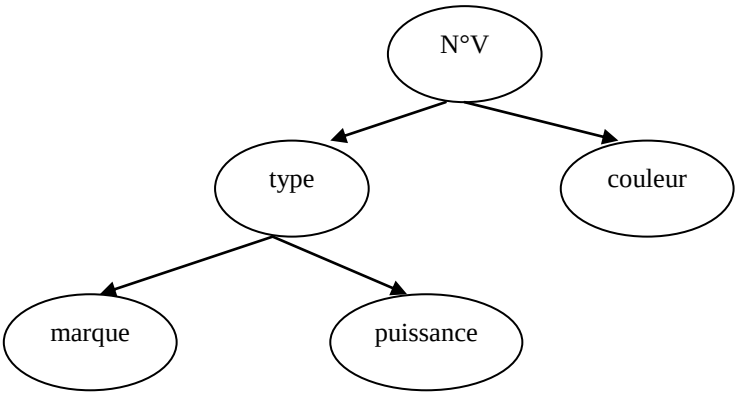
D. Graphe des dépendances fonctionnelles

C'est une représentation graphique des dépendances fonctionnelle élémentaire.

Exemple 6:
On considère les dépendances fonctionnelles de la relation voiture :

F={N°V→type, type→marque, type→puissance, N°V→couleur}

La figure suivante représente le graphe associé aux dépendances fonctionnelles de F



FIGI.8. Graphe des dépendances fonctionnelles élémentaires

E. Fermeture transitive et couverture minimale

A partir d'un ensemble de DF élémentaires, on peut composer par transitivité d'autres DF élémentaires. On aboutit ainsi à la notion de fermeture transitive d'un ensemble F de DF élémentaires.

F. Notion de fermeture transitive

C'est l'ensemble des DF élémentaires considérées enrichis de toutes les DF élémentaires déduites par transitivité.

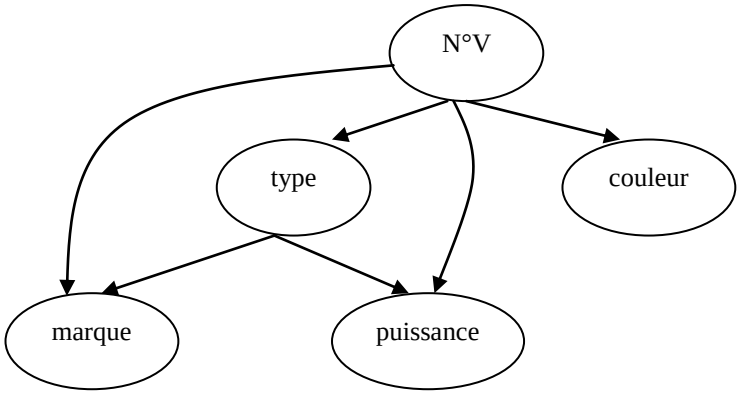
Exemple7 : à partir de l'ensemble de DF :

F={N°V→type, type→marque, type→puissance, N°V→couleur}

On déduit la fermeture transitive :

$F+=F^{Y}$ {N°V→marque, N°V→puissance}

Le graphe correspondant à fermeture transitive de F est :



FIGI.9. Graphe correspondant à fermeture transitive de F

A partir de la notion de fermeture transitive, il est possible de définir l'équivalence de deux ensembles de DF élémentaires : deux ensembles sont équivalents s'ils ont la même fermeture transitive.

D'autre part, il est intéressant de déterminer un sous-ensemble minimum de DF permettant de générer toutes les autres DF.

G. Notion de couverture minimale

C'est un ensemble de DF élémentaires associé à un ensemble d'attributs vérifiant les propriétés suivantes.

- Aucune DF n'est redondante.
- Toute DF élémentaire des attributs est dans la fermeture transitive.

Exemple 8 :

Soit F l'ensemble des DF élémentaires :

F={N°V→type, type→marque, type→puissance, N°V→couleur}. F est une couverture minimale pour l'ensemble des DF de la base de données voiture.

La couverture minimale est essentielle pour décomposer les relations sans perte d'informations.

5. Notion de clé et trois premières formes normales

A. Notion de clé d'une relation

La clé d'une relation c'est un sous-ensemble X des attributs d'une relation R(X1, X2, ….,Xn) telque

1-A → X1, X2, ….,Xn

2-Il n'existe pas de sous-ensemble B⊂ A telque B → X1, X2, ….,Xn

Exemple9:

N°SS est une clé de la relation Personne.

B. Notion des trois premières formes normales

Les trois premières formes normales ont pour objectif de supprimer toutes les redondances et les anomalies dans la base de données et de permettre la décomposition de relations en plusieurs sous relations sans perdre d'informations, à partir de la notion de dépendance fonctionnelle.

B.1. Définition de la première forme normale (1FN)

Une relation R est en première forme normale, si tout attribut contient une valeur atomique.

Cette forme normale consiste simplement à éviter les domaines composés de plusieurs valeurs.

Exemple 10:

la relation personne (nom, prénom) sera ainsi décomposée en personne1 (nom, prenom1) et personne2 (nom, prenom2).

B.2. Définition de la deuxième forme normale (2FN)

Une relation R est en deuxième forme normale si et seulement si :

1. elle est en première forme normale
2. chaque attribut non clé, dépend fonctionnellement de la clé dans sa totalité

Exemple 11:

Soit la relation suivante :

prod_fourn(N°prod, N°fourn, nom_fourn, désignation_prod, prix_achat, prix_vente, N°acheteur, nom_acheteur, taux_reduction)

Nous remarquons qu'il existe une anomalie dans la relation **prod_fourn**, en effet la 2èmeFN n'est pas vérifiée :

Il existe des attributs dépend de certain attribut de clé (pas de totalité)

La solution est d'éclater la relation **prod_fourn** en plusieurs relations en 2FN :

fournisseur(N°fourn, nom_fourn)

produit(N°prod, désignation_prod, prix_vente, N°acheteur, nom_acheteur)

achat(N°prod, N°fourn, prix_d'achat, taux_réduction)

B.3. Définition de la troisième forme normale (3FN)

Une relation R est en troisième forme normale, si et seulement si :

1. elle est en deuxième forme normale
2. tous attributs n'appartenant pas à une clé ne dépendent pas d'un attribut non clé.

Exemple:

Dans l'exemple précédent on remarque que la relation produit n'est pas en 3FN car :

N°acheteur $\rightarrow$ nom_acheteur

On va éclater produit en deux relations :

produit(N°prod, désignation_prod, prix_vente, N°acheteur)

acheteur(N°acheteur, nom_acheteur)

B.4. Définition (Forme normale de BOYCE-CODD) :

Une relation est en Forme normale de BOYCE-CODD (BCNF) si, et seulement si, les seules dépendances fonctionnelles élémentaires sont celles dans lesquelles une clé détermine un attribut.

6. Des exercices d'applications:

EXERCICE N°01:

Quels sont les avantages du modèle relationnel?

SOLUTION:

1. **SIMPLICITE DE PRÉSENTATION**
   Représentation sous forme de tables

2. **OPÉRATIONS RELATIONNELLES**
   Algèbre relationnelle
   Langages assertionnels

3. **INDEPENDANCE PHYSIQUE**
   Optimization des accès
   Stratégie d'accès déterminée par le système

4. **INDEPENDANCE LOGIQUE**
   Concept de VUES

5. **MAINTIEN DE L'INTEGRITÉ**
   Contraintes d'intégrité définies au niveau du schéma

EXERCICEN°02:

A partir de deux domaines suivant déduire les relations équivaut de ces domaines?

Les domaines :

NOM_ETUD = { seddiki, benahmed }

PREN_ETUD = { noureddine, khélifa, mohamed }

DATE_NAISS = {Date entre 1/1/1970 et 31/12/1960}

NOM_SPORT = { judo, tennis, foot }

SOLUTION:

R ⊂ D1 × D2 × ... × Dn

D1, D2, ... , Dn sont les domaines de R n'est le degré ou l'arité de R

La relation ETUDIANT

ETUDIANT ⊂ NOM_ETUD × PREN_ETUD × DATE_NAISS

EUDIANT = { (seddiki, noureddine, 1/1/1972), (durant, jacques, 2/2/1964) }

La relation INSCRIPT

INSCRIPT ⊂ NOM_ETUD× NOM_SPORT

INSCRIPT = { seddikit, judo), (seddiki, foot), Benahamed, judo) }

EXERCICEN°03:

Soit la relation suivante:

EMPLOYEE (Id_Emp , nomEmp, prénomEmp, adresse, voiture)

Es que la relation EMPLOYEE est en 1FN sinon normalisé la?

SOLUTION:

Cette relation n'est pas en 1FN, car l'attribut adresse est composite et l'attribut voiture est multiple.
Il faut le décomposer en deux relations:

EMPLOYEE (Id_Emp , nomEmp, prénomEmp, numéroEtRue, codePostal, ville)

VOITURE(idVoiture, modèle, marque, propriétaire)
Où propriétaire est une clé étrangère qui fait référence au schéma relation EMPLOYEE

Remarque :

La première forme normale impose que chaque ligne d'une relation ait une seule valeur pour chaque colonne (ou attribut), ce qui est justement la définition d'une table Donc, une table est nécessairement en première forme normale au sens du modèle relationnel.

EXERCICE N°04:

Soit le schéma de relation suivant :

AFFECTATION(idPersonne, dEtablissement, nomPersonne, prénomPersonne, nomEtablissement)

Supposons qu'une personne puisse être affectée à plusieurs établissements et qu'à un établissement sont affectées plusieurs personnes.

Es que cette relation est en 2FN sinon normalisé la ?

SOLUTION:

Cette relation n'est pas en 2FN car, par exemple, nomPersonne ne dépend que de idPersonne. Pour normaliser cette relation, il faut la décomposer de la manière suivante :

PERSONNE(idPersonne, nomPersonne, prénomPersonne)

ETABLISSEMENT(idEtablissement, nomEtablissement)

AFFECTATION(idPersonne, idEtablissement)

Où idPersonne et idEtablissement sont des clés étrangères qui font respectivement référence aux schémas de relation PERSONNE et ETABLISSEMENT.

EXERCICEN°05:

Soit le schéma relationnel suivant :

STAGIAIRE (idStg, nomStg, prénomStg, établissement, villeEtablissement)

Es que cette relation est en 3FN sinon normalisé la ?

SOLUTION:

Cette relation n'est pas 3FN, car l'attribut villeEtablissement dépend de établissement qui n'est pas une clé candidate. Pour normaliser cette relation, il faut la décomposer de la manière suivante :

STAGIAIRE (idStg, nomStg, prénomStg, établissement, idEtablissement) où idEtablissement est une clé étrangère qui fait référence au schéma ETABLISSEMENT.

ETABLISSEMENT (idEtablissement, nomEtablissement, villeEtablissement).

Exercice supplémentaire:

Soit la relation suivante :
Employé (N°emp , nom_emp , adresse , catégorie , salaire)
Avec les dépendances fonctionnelles suivantes :
N°emp ⟶ nom_emp , adresse , salaire , catégorie
catégorie ⟶ salaire

1. Quelle est la clé primaire de la relation Employé
2. Dans quelle forme normale est la relation Employé
3. Décomposer la en 3FN si elle ne l'est pas déjà.

7. COCLUSION

Dans ce chapitre, nous avons présenté les deux modèles de données, réseau et hiérarchique, qui ont existé bien avant le modèle relationnel. Compte tenu de l'accroissement des possibilités des ordinateurs, constaté à partir des années70, ces modèles sont actuellement critiqués. Ils peuvent rester très compétitifs au niveau interne. Les critiques proviennent surtout de la complexité des langages de manipulations associés à ces modèles, basé sur la 'navigation', donc peu accessibles aux utilisateurs néophytes. Et nous avons présenté encore les concepts de base de modèle relationnelle.

Chapitre II
Les bases de données et les objets

II .Les bases de données et les objets

1. Introduction

Aujourd'hui, en programmation, il existe deux principaux modèles de représentation du monde :

- Le modèle fonctionnel
- Le modèle objet

Dans une approche fonctionnelle, les programmes sont composés d'une série de fonctions, qui assurent ensemble certains services. Il s'agit d'une approche logique, cohérente et intuitive de la programmation.

Cette approche a un avantage certain appelé la factorisation des comportements (c'est à dire que pour créer une fonction d'une application, rien ne vous empêche d'utiliser un autre ensemble de fonctions (qui sont donc déjà écrites)).

Mais, l'approche fonctionnelle a aussi ses défauts, comme par exemple une maintenance complexe en cas d'évolution d'une application (une simple mise à jour de l'application à un point donné peut impacter en cascade sur d'autres fonctions de l'application). L'application sera alors retouchée dans sa globalité.

L'approche fonctionnelle n'est pas adaptée au développement d'applications qui évoluent sans cesse et dont la complexité croit continuellement (plusieurs dizaines de milliers de lignes de code).

L'approche objet possède donc plusieurs avantages :

- La modélisation des objets de l'application (consiste à modéliser informatiquement un ensemble d'éléments d'une partie du monde réel en un ensemble d'entités informatiques. Ces entités informatiques sont appelées objet).
- La modularité (la programmation modulaire permet la réutilisation du code, via l'écriture de librairies).
- La réutilisabilité.
- L'extensibilité (pour une meilleure productivité des développeurs et une plus grande qualité des applications).

- L'approche objet a été donc inventée pour faciliter l'évolution d'applications complexes. Elle apporte l'indépendance entre les programmes, les données et les procédures parce que les programmes peuvent partager les mêmes objets sans avoir à se connaître comme avec le mécanisme d'import/export.

- L'approche objet a introduit indépendamment de tout langage de programmation à l'objet trois concepts de base : objet, classe et héritage entre classes. Les concepts objet et classe sont interdépendants, c'est-à-dire qu'un objet est une instance d'une classe et la classe décrit la structure et le comportement communs d'objets (ses instances).

2. Atouts du relationnel

- Très bon support théorique
- Simplicité des concepts basés sur des ensembles
- Modèle de données (externe, logique, physique)
- LDD et LMD

3. Atouts d'un SGBD(R)

- Persistance : les données survivent aux programmes.
- Taille : accès efficace aux grandes quantités de données : gestion des tampons, index, optimisation des requêtes.
- Performances sur le transactionnel (OLTP).
- Standard industrie :
    a. SQL2 : indépendance des applications
    b. Architecture client-serveur
    c. Optimisation de requêtes

4. Limite du relationnel

- Modèle plat : pas de données complexes
    - Difficile d'exprimer des choses structurées
        - ✓ 1FN
    - Nombre des relations
    - Difficile de naviguer dans la structure
        - ✓ Jointure
        - ✓ inadaptés aux application navigationnelles (réseaux, CAO)
    - Pas de types nouveaux
        - ✓ Volumineux ou/et multimédia
    - Non intégration des opérations

- SQL déclaratif, simple
    - Langage non complet

- ✓ interfaçage (connexion) avec un langage de programmation.

5. Définition d'un objet

Entité informatique complète identifiable caractérisée pas son état et un ensemble de méthodes.

Associe traitements (méthodes) et données dans une même entité.

L'interface est le seul moyen d'accéder à l'objet, par l'envoi de messages.

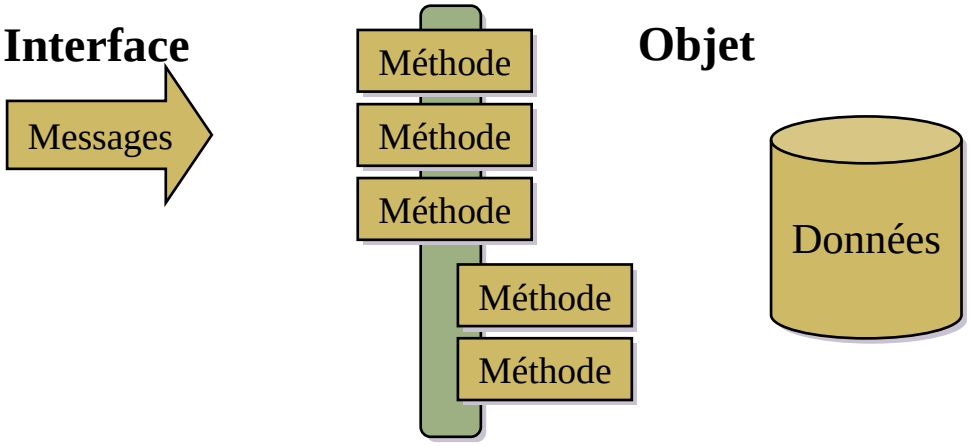


FIG.II.1. Interface Objet

L'abstraction informatique d'une entité du monde réel caractérisée par une identité, un état et un comportement.

Un objet est l'instance d'une classe, et qui est un type de données abstrait, caractérisé par des propriétés (ses attributs et ses méthodes) communes à des objets, qui permet de créer des objets possédant ces propriétés.

Objet = identité + état (attributs) + comportement (méthodes)

- Chaque objet a une identité indépendante de sa valeur.
- On peut mettre à jour la valeur sans changer l'identité.
- Identité et égalité (surface ou profondeur?) sont deux concepts différents.
- Les objets peuvent être partagés.
- Les objets peuvent être périodiques.
- Les identificateurs ne sont pas accessibles aux utilisateurs.

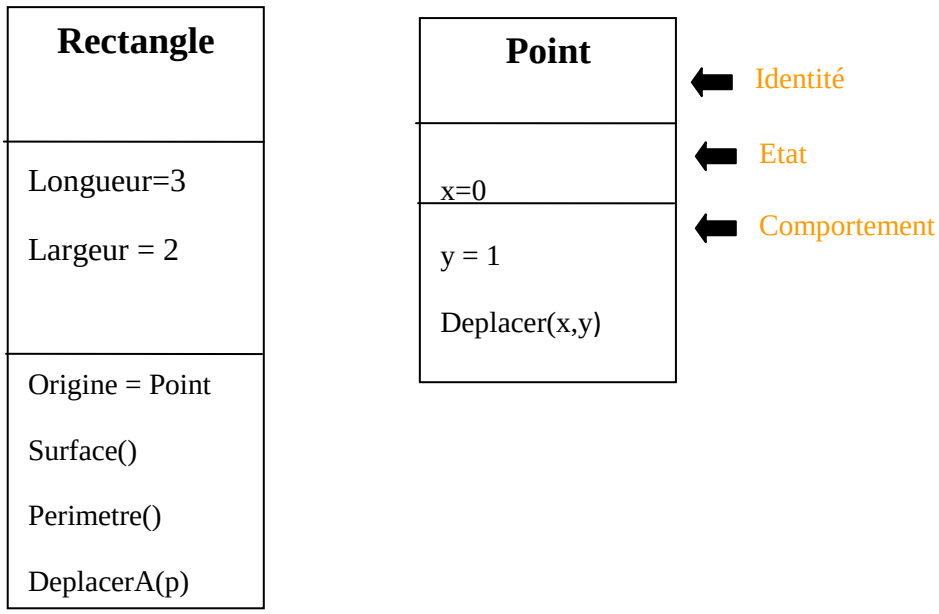
A. Les concepts de base d'un objet

- **L'identité :** L'objet possède une identité, qui permet de le distinguer des autres objets, indépendamment de son état. On construit généralement cette identité grâce à un identifiant découlant naturellement du problème (par exemple un produit pourra être repéré par un code, une voiture par un numéro de série, ...). L'identifiant (OID - Object Identifier ou PID - Persistent Identifier) permet de distinguer chaque objet de la base. L'identifiant ne doit pas être géré par le programmeur mais par le système. Un objet est un couple (oid, valeur) tel que: **(oid1, [nom: N, prénom: P, Enfants: {E1, E2, E3}])**

- **Les attributs :** Il s'agit des données caractérisant l'objet. Ce sont des variables stockant des informations d'état de l'objet

- **Les méthodes (appelées parfois fonctions membres):** Les méthodes d'un objet caractérisent son comportement, c'est-à-dire l'ensemble des actions (appelées opérations) que l'objet est à même de réaliser. Ces opérations permettent de faire réagir l'objet aux sollicitations extérieures (ou d'agir sur les autres objets). De plus, les opérations sont étroitement liées aux attributs, car leurs actions peuvent dépendre des valeurs des attributs, ou bien les modifier.

B. Exemple



C. Modèle objet pur

- Premier SGBD objet : 1983 (Gemstone)

- Approche : étendre le langage de programmation objet aux fonctions de SGBD
- Persistance
    - Orthogonale au type
    - langages : C++, smalltalk, Java/OQL
- Produits
- O2, ObjectStore, Ontos, Objectivity, Jasmine, Versant
- Niches technologiques
    - réseau, CAO, SIG, Gestion de données Techniques
    - pas de transactionnel lourd

D. Règles d'or d'un SGBD Objet

Pour être qualifié d'objet, un SGBD doit respecter treize règles :

1. Données persistantes
    - Les données persistantes et non persistantes (temporaires) sont manipulées de la même façon par un programme.
2. Grande quantité de données
    - Techniques de regroupement physique, d'indexation, d'optimisation de requêtes et de gestion de cache.
3. Fiabilité des données
    - Transactions et privilèges
4. Partages de données
    - multi-utilisateur → mécanismes de verrous
5. Facilité d'interrogation
    - Langage de requêtes.
    - Le résultat d'une requête n'est pas forcément un objet d'une classe existante.
    - Le langage offre en outre la puissance d'un langage de programmation
6. Encapsulation
    - Les données sont accessibles par des méthodes à différents degrés (visibilité)
7. Objets composites
    - Les structures de données à définir et à manipuler peuvent être complexes
8. Identificateur d'objet
    - Accès aux objets directement ou par des liens inter-objets via les OID
    - Objets partagés
9. Classes, types et méthodes
    - Types abstraits de données
    - Concepts de classes et de méthodes de l'approche objet
10. Héritage
11. Surcharge et liaison dynamique

- Les méthodes peuvent être surchargées. La liaison dynamique est la capacité d'établir la correspondance entre le nom d'une méthode et son implantation lors de l'exécution et non lors de la compilation

12. Langage de programmation complet
    - Le système dispose d'un langage de programmation qui ne nécessite pas d'opérateur externe pour écrire une application
13. Extensibilité
    - Le système autorise l'ajout dynamique de nouveaux types abstraits de donnée, de nouvelles classes, de nouvelles méthodes, etc.

E. SGBDO ++

- Héritage multiple
- Gestion des versions
- CI (contrainte d'intégrité)
- Vues
- Evolution de schéma
- Persistance dynamique

6. Atouts de la technologie objet dans les bases de données

- Origine dans les langages de programmation objet
- Objectif principal de l'approche objet : augmenter le niveau d'abstraction.
- Objet dans les SGBD : combiner les avantages
    - fichiers (simplicité et rapidité d'accès),
    - bases hiérarchiques ou réseaux (navigation entre objets)
    - bases relationnelles (langage de requêtes)
- Réduction des différences
    - entre le langage de programmation et la BD,
    - entre le monde à modéliser et le modèle de données d'autre part

7. Limites de la technologie objet dans les bases de données

- Manque de théorie dans la conception d'un schéma, modèle de données:
    - complexe
    - manque de philosophie

8. SGBD objet-relationnel

- Un SGBD objet-relationnel doit prendre en compte les quatre mécanismes suivants:
    - L'extension des types de données
    - Les objets complexes (en terme de structures de données)
    - L'héritage
    - Les systèmes de règles

9. Propriétés RICE (Réutilisation, Identité, Complexité et Encapsulation)

- Réutilisation
    - finalité du paradigme objet
    - héritage, généralité, composition, polymorphisme
- Identité
    - identifier un objet de manière unique
- Complexité
    - définition d'objets complexes et/ou fortement structurés
- Encapsulation
    - boîte noire avec des méthodes de manipulation

10.RICE/BDO/BDOR

Voila un table comparative entre BDO et BDOR selon la critère RICE

	BDO	**BDOR**
Réutilisation	Héritage multiple	Héritage ? Polymorphisme (surcharge)
Identité	OID	ROWID
Complexité	Collections (SET, BAG, LIST, ARRAY)	ADT Collections (SET, LIST, MULTISET) Operations VALUE
Encapsulation	Attributs SET_VALUE et GET_VALUE Méthodes	FONCTION et PROCEDURE

Tab.II.1. RICE/BDO/BDOR

11.Les messages ou la communication entre objets

1. Par rapport aux messages on distingue trois catégories d'objets :

    - les acteurs (qui envoient des messages)
    - les serveurs (qui attendent des messages)
    - les agents (qui savent envoyer et recevoir des messages)

2. Il y a 5 types de messages :

    - **les constructeurs** : qui spécifient comment créer un objet .
    - **les destructeurs :** qui spécifient ce qui se passe quand un objet disparait.
    - **les sélecteurs :** qui renvoient tout ou partie de l'état d'un objet
    - **les modificateurs :** qui changent l'état de l'objet .
    - **les itérateurs :** qui appliquent un traitement sur une liste d'attributs.

3. On distingue aussi les messages asynchrones, synchrones, retardés et minutés.

Il est important de noter qu'on distingue deux types de comportement d'objet en réponse à un message :

- **statique** (si le traitement du message ne dépend pas de d'autres paramètres).
- **dynamique** (si le traitement du message dépend de paramètres externes ou internes à l'objet).

La plupart des objets ont un comportement mixte :

Statiques pour certains messages, dynamiques pour d'autres.

12.Les classes

On appelle classe la structure d'un objet, c'est-à-dire la déclaration de l'ensemble des entités qui composeront un objet. Les objets de même nature ont en général la même structure et le même comportement. La classe factorise les caractéristiques communes de ces objets et permet de les classifier. Un objet est donc, une instanciation d'une classe, c'est la raison pour laquelle on pourra parler indifféremment d'objet ou d'instance (éventuellement d'occurrence). Toutes les instances d'une classe constituent l'extension de la classe.

La classe est un type abstrait de données caractérisée par des propriétés (attributs et opérations) communes à ses objets, et un mécanisme permettant de créer des objets ayant ces propriétés.

Classe = instanciation + attributs (variables d'instances)+ opérations

- L'instanciation : L'objet possède une identité, qui permet de le distinguer des autres objets, indépendamment de son état. L'instanciation représente la relation entre un objet et sa classe d'appartenance qui a permis de le créer.
- Les attributs (appelés aussi variables d'instances): Ils ont un nom et soit un type de base (simple ou construit) soit une classe (l'attribut référence un objet de la même ou une autre classe).
- Les opérations (appelées parfois méthodes): Elles sont les opérations applicables à un objet de la classe. Elles peuvent modifier tout ou en partie l'état d'un objet et retourner des valeurs calculées à partir de cet état.

classe = instanciation + attributs (variables d'instances)+ opérations

- L'instanciation : L'objet possède une identité, qui permet de le distinguer des autres objets, indépendamment de son état. L'instanciation représente la relation entre un objet et sa classe d'appartenance qui a permis de le créer.

- Les attributs (appelés aussi variables d'instances): Ils ont un nom et soit un type de base (simple ou construit) soit une classe (l'attribut référence un objet de la même ou une autre classe).
- Les opérations (appelées parfois méthodes): Elles sont les opérations applicables à un objet de la classe. Elles peuvent modifier tout ou en partie l'état d'un objet et retourner des valeurs calculées à partir de cet état.

A. L'héritage entre classes

L'héritage est un principe propre à la programmation orientée objet, permettant de créer une nouvelle classe à partir d'une classe existante. Le nom d'"héritage" (ou parfois dérivation de classe) provient du fait que la classe dérivée (la classe nouvellement créée) contient les attributs et les méthodes de sa superclasse (la classe dont elle dérive).

L'intérêt majeur de l'héritage est de pouvoir définir de nouveaux attributs et de nouvelles méthodes pour la classe dérivée, qui viennent s'ajouter à ceux et celles héritées. Par ce moyen on crée une hiérarchie de classes de plus en plus spécialisées.

L'héritage est un mécanisme de transmission des propriétés d'une classe vers une sous-classe.

La hiérarchie des classes ou l'association (relationship) est un concept essentiel pour la modélisation des données. On peut représenter sous forme de hiérarchie de classes, parfois appelée arborescence de classes, la relation de parenté qui existe entre les différentes classes. La hiérarchie commence par une classe générale appelée superclasse (classe de base, classe parent, classe ancêtre, classe mère ou classe père). Puis les classes dérivées (classe fille ou sous-classe) deviennent de plus en plus spécialisée.

B. Exemple d'héritage

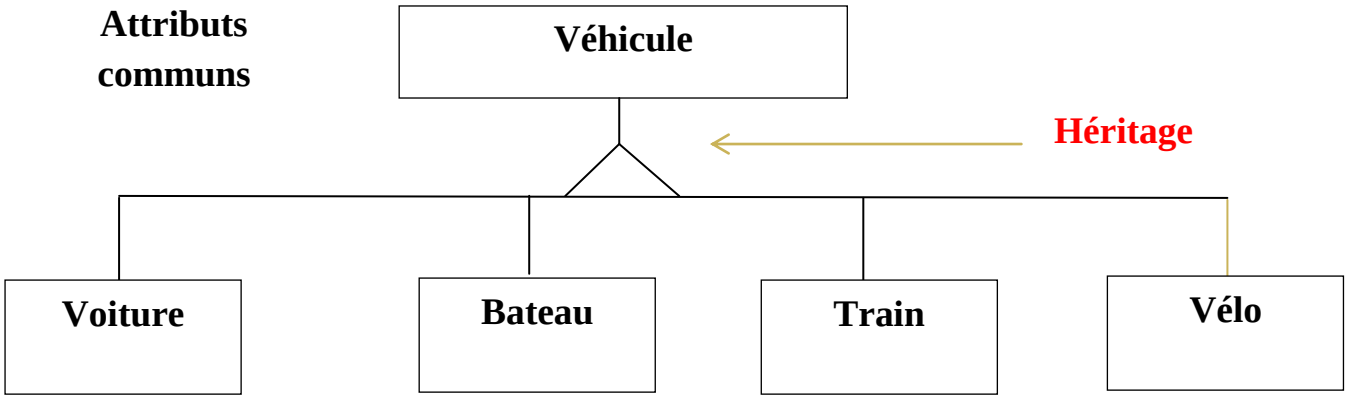


FIG.II.2.Exemple d'héritage

13.Notion de polymorphisme

Le nom de polymorphisme vient du grec et signifie qui peut prendre plusieurs formes. Cette caractéristique essentielle de la programmation orientée objet caractérise la possibilité de définir plusieurs fonctions de même nom mais possédant des paramètres différents (en nombre et/ou en type), si bien que la bonne fonction sera choisie en fonction de ses paramètres lors de l'appel.

Le polymorphisme représente la faculté d'une opération de s'appliquer à des objets de classes différentes.

Le polymorphisme rend possible le choix automatique de la bonne méthode à adopter en fonction du type de donnée passée en paramètre (par exemple l'utilisation de la méthode addition():

- int addition(int, int) pourra retourner la somme de deux entiers.
- float addition(float, float) pourra retourner la somme de deux flottants.
- char addition(char, char) pourra définit au gré de l'auteur la somme de deux caractères.

  Le polymorphisme nécessite une liaison dynamique des messages à des méthodes. La liaison dynamique se produit lorsque la décision concernant la méthode à exécuter se prend lors de l'exécution d'un programme. Elle est nécessaire lors que:

  - Une variable réfère à un objet dont la classe d'appartenance fait partie d'un arbre d'héritage
  - Et lorsque plusieurs méthodes existent pour le même message dans l'arbre d'héritage.

14.Encapsulation

L'encapsulation (ou encapsulage) en général est la notion de mettre une chose dans une autre. En imageant, on peut voir que cette chose est mise dans une capsule. En particulier, on peut retrouver ce terme dans plusieurs domaines en programmation, l'encapsulation de données est l'idée de cacher l'information.

En programmation, l'encapsulation désigne le principe de regrouper des données brutes avec un ensemble de routines permettant de les lire ou de les manipuler. Ce principe est souvent accompagné du masquage de ces données brutes afin de s'assurer que l'utilisateur ne contourne pas l'interface qui lui est destinée. L'ensemble se considère alors comme une boîte noire ayant un comportement et des propriétés spécifiés.

L'encapsulation est un pilier de la programmation orientée objet, où chaque classe définit des méthodes ou des propriétés pour interagir avec les données membres, mais ce principe peut se rencontrer dans d'autres styles de programmation (par exemple la programmation modulaire).

15.Agrégation

L'agrégation est un type de relation entre les objets. Elle permet de décrire un objet par les objets qui le composent.

16.Extensibilité

Un SGBDOO doit supporter la possibilité d'ajouter de nouveaux types afin de prendre en compte de nouveaux domaines d'application.

17.Structure de données

Les objets sont décrits par des attributs, et sont regroupés en classes.

Chaque objet a une identité qui le distingue de tous les autres. On l'appelle oid de l'objet (en anglais "object identity").

À ce jours, chaque SGBDO possède son propre modèle de données OO. Il n'y a pas encore de norme. Nous présentons ici les points essentiels sur lesquels il y a consensus:

A. Structure complexe

La structure d'une classe est définie en utilisant des constructeurs comme: TUPLE, SET.

Le constructeur « TUPLE » crée un type tuple composé d'une suite d'attributs:

TUPLE (att1:dom1, … attn: domn) où domi peut être un domaine prédéfini

(STRING, REAL, INT, DATE…) comme il peut être défini par un constructeur (TUPLE, SET) ou bien peut être un nom d'une classe.

Le constructeur d'ensemble « SET » crée un type ensemble composé d'un ensemble de valeurs:

SET domaine

Où domaine à la même définition précédente.

Certains SGBDO proposent des constructeurs de collection autres que l'ensemble, tels que la liste, le multi-ensemble, le tableau, ...

Exemple :

```
CLASS Etudiant {
      num : INT,
      nom : STRING,
      prénoms : SET STRING,
     date-nais : DATE,
     adresse : TUPLE (n° : INT,
          rue : STRING,
          ville : STRING,
         pays : STRING)
    cours-suivis : SET TUPLE ( nom-cours : STRING,
          note : REAL)
}
```

A.1. Liens de composition

Un lien de composition relie une classe C1 à une classe C2 si les objets de C1 (appelés "objets composites") sont composés d'objets de C2 (appelés "objets composants").

Exemple :

```
CLASS Etudiant {
     num : INT,
     nom : STRING,
    prénoms : SET STRING,
   cours-suivis : SET TUPLE (cours : Cours,
      note : REAL) }


CLASS Cours {
    nomC : STRING, .... }
```

La classe Cours est une classe composante de la classe Etudiant

A.2. Identité d'objet

L'identifiant doit être permanent, fixe (ne change pas pendant la vie de l'objet) et unique dans la base et dans le temps (deux objets différents dans une base n'auront jamais le même identifiant, même s'ils n'existent pas en même temps).

Ces caractéristiques de l'identifiant permettent à plusieurs objets de partager le même objet composant.

On peut dire que deux objets sont égaux dans trois cas:

• L'égalité d'identité

obj1 == obj2

Signifie : « le même oid »

• L'égalité de surface

obj1 =1 obj2

Signifie: « les objets obj1 et obj2 ont la même valeur au premier niveau » C'est-à-dire tous leurs attributs ont la même valeur, que ce soient des attributs-valeur ou des attributs-référence.

18.Graphe de généralisation des classes

La notion d'héritage permet de définir deux types de classes: classe CS (Classe Spécifique) d'une autre classe CG (Classe Générique), notée:

CS Is-a CG

CS représente un sous-ensemble de CG. Du point de vue conceptuel, l'ensemble des objets de CS est inclut dans celui de CG;

Exemple :

Soit le graphe de généralisation des classes décrivant les personnes dans une université.

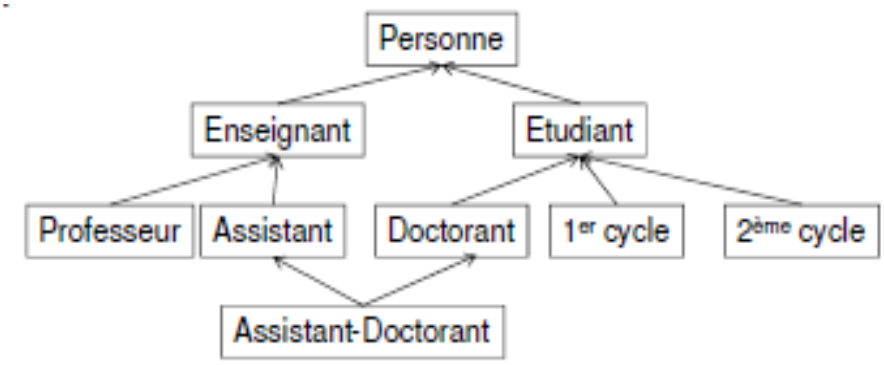


FIG.II.3.Exemple d'héritage

19.Le modèle de données

- Le modèle de données objet-relationnel est une extension du modèle de données relationnel.
- Une des principales extensions consiste à manipuler des structures de données complexes incluant.
  - ✓ Des pointeurs (facilitent la navigation).
  - ✓ Des tables imbriquées (permettent de s'affranchir de la règle de la première forme normale).

- Les objets sont stockés dans des tables objets-relationnelles qui présentent des caractéristiques semblables au modèle de données

20.Relationnel vs. Objet / le modèle relationnel

POUR	CONTRE
• Le modèle relationnel est adapté à un grand nombre d'applications (de gestion par exemple), qui utilisent des types simples de données. • Le modèle relationnel est basé sur une théorie "scientifique" et a été mis à l'épreuve ces 15 dernières années. • Le modèle relationnel est facile à comprendre. • Les objets permettent de représenter [plus] naturellement des entités du monde réel: o objets complexes. • Les objets offrent une plus grande souplesse pour gérer les structures de données complexes (multimedia, CAO, AGL, etc.)	• Le modèle relationnel ne permet pas de représenter certaines entités complexes du monde réel. • Les jointures peuvent être des opérations relativement lourdes. • • Les SGBDO posent un certain nombre de problèmes: o gestion des objets en mémoire et dans la base; o performance d'accès aux objets; o concurrence d'accès aux objets.

Tab.II.2. Relationnel vs. Objet / le modèle relationnel

21.Conclusion

Dans ce chapitre, nous avons présenté le modèle orientée objet, cette approche à été inventée pour faciliter l'évolution d'applications complexes. Elle apporte l'indépendance entre les programmes, les données et les procédures parce que les programmes peuvent partager les mêmes objets sans avoir à se connaître comme avec le mécanisme d'import/export.

Chapitre III
Les bases de données Réparties

III. Les Bases de Données Réparties

1. Introduction

L'évolution des techniques informatiques depuis les vingt dernières années permet d'adapter les outils informatiques à l'organisation des entreprises (et non l'inverse). La puissance des micro-ordinateurs et des stations de travail, la fiabilité et la souplesse des SGBD relationnels, les performances des réseaux permettent d'envisager une répartition des ressources informatiques tout en préservant l'essentiel, c'est à dire la cohérence des bases de données.

2. Problématique

Les pressions pour la distribution :

- Il devient impératif de décentraliser l'information (cas des multinationales),
- Augmentation du volume de l'information (14 fois de 1990 à 2000),
- Augmentation du volume des transactions (10 fois dans les 5 prochaines années).

3. Les Besoin d'une répartition de données :

A. Besoin de serveurs de BDs qui fournissent un bon temps de réponse sur des gros volumes de données.

A.1. Prédiction d'accroissement:

- Vitesse des microprocesseurs : 50% par an,
- Capacité des DRAM : 4 fois tous les 4 ans,
- Débit des disques : 2 fois sur les 10 dernières années.

B. Passage d'étranglement sur les E/Ss.

B.1. Pour améliorer le débit des E/Ss :

- Partitionnement des données,
- Accès parallèle aux données,
- Utiliser plusieurs nœuds (avec un bon coût/ performance), et les faire communiquer par un réseau.
- Les BDRs se sont développées, grâce au progrès technologiques réalisés au niveau de l'infrastructure réseau et des postes de travail.

4. Buts de la répartition des bases de données

Les bases de données réparties ont une architecture plus adaptée à l'organisation des entreprises décentralisées.

Plus de fiabilité : les bases de données réparties ont souvent des données répliquées (répartir). La panne d'un site n'est pas très importante pour l'utilisateur, qui s'adressera à autre site.

- Meilleures performances : réduire le trafic sur le réseau est une possibilité d'accroître les performances. Le but de la répartition des données est de les rapprocher de l'endroit où elles sont accédées. Répartir une base de données sur plusieurs sites permet de répartir la charge sur les processeurs et sur les entrées/ sorties.

- Faciliter l'accroissement (l'augmentation) : l'accroissement se fait par l'ajout de machines sur le réseau.

5. SGBD réparti

Une base de données centralisée est gérée par un seul SGBD, est stockée dans sa totalité à un emplacement physique unique et ses divers traitements sont confiés à une seule et même unité de traitement. Par opposition, une base de données distribuée est gérée par plusieurs processeurs, sites ou SGBD.

Un système de bases de données réparties ne doit donc en aucun cas être confondu avec un système dans lequel les bases de données sont accessibles à distance. Il ne doit non plus être confondu avec une multi base ou une BD fédérée.

Dans une multi base, plusieurs BDs interrogèrent avec une application via un langage commun et sans modèle commun.

Dans une BD fédérée, plusieurs BDs hétérogènes sont accédées comme une seule via une vue commune.

Du point de vue organisationnel nous distinguons deux architectures :

A. **Architecture Client-Serveur :** les serveurs, ont pour rôle de servir les clients. Par servir, on désigne la réalisation d'une tâche demandée par le client.
B. **Architecture Pair-à-Pair (Peer-to-Peer, P2P) :** par ce terme on désigne un type de communication pour lequel toutes les machines ont une importance équivalente.

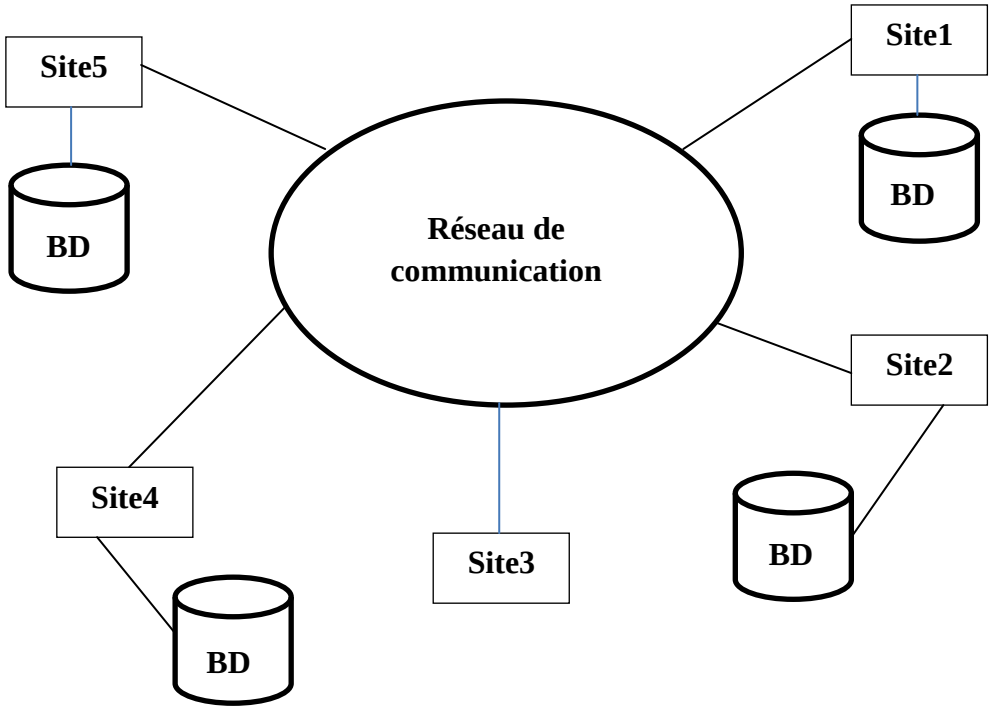


FIG.III.1. SGBD Répartie

6. Objectifs définis par Christopher .J. Date

Les principaux objectifs sont:

- Transparence pour l'utilisateur
- Autonomie de chaque site
- Absence de site privilégié
- Continuité de service
- Transparence vis à vis de la localisation des données
- Transparence vis à vis de la fragmentation
- Transparence vis à vis de la réplication
- Traitement des requêtes distribuées
- Indépendance vis à vis du matériel
- Indépendance vis à vis du système d'exploitation
- Indépendance vis à vis du réseau

- Indépendance vis à vis du SGBD

7. Problèmes à surmonter

- **Coût :** la distribution entraîne des coûts supplémentaires en terme de communication, et en gestion des communications (hardware et software à installer pour gérer les communications et la distribution).
- **Problème de concurrence.**
- **Sécurité :** la sécurité est un problème plus complexe dans le cas des bases de données réparties que dans le cas des bases de données centralisées.

8. Définition

Une base de données répartie (BDR) est un ensemble structuré et cohérent de données, stocké sur des processeurs distincts, et géré par un système de gestion de bases de données réparties (SGBDR).

Une base de données distribuée ou répartie est une base de données dont les différentes parties sont stockées sur des sites (géographiquement distants), reliés par un réseau. La réunion de ces parties forme la base de données distribuée.

9. Types de BDR

A. BDR homogène : Obtenue en divisant une BD en un ensemble de BD locales, chacune étant gérée par le même SGBD, même modèle de données et même langage de requêtes.

Exemple : DB2, ORACLE (SQL)

B. BDR Hétérogènes : Deux niveaux d'hétérogénéité :

- Les BD ont le même modèle (relationnel) mais sont gérées par des SGBD différents (Oracle, SQL server, ….).
- Les BD ont des modèles différents (relationnel, objet) et gérées par des SGBD différents (Oracle, O2).

C. BDR hétérogène : BD répartie obtenue en intégrant dans une BD unique un ensemble de BD locales gérées par des SGBD différents.

10.Conception d'une base de données répartie

La définition du schéma de répartition est la partie la plus délicate de la phase de conception d'une BDR car il n'existe pas de méthode exacte pour trouver la solution optimale. L'administrateur doit donc prendre des décisions en fonction de critères techniques et organisationnels avec pour objectif de minimiser le nombre de transferts entre sites, les temps de transfert, le volume de données transférées, les temps moyens de traitement des requêtes, le nombre de copies de fragments, etc...

A. Conception descendante (top down design)

On commence par définir un schéma conceptuel global de la base de données répartie, puis on distribue sur les différents sites en des schémas conceptuels locaux.

La répartition se fait donc en deux étapes, en première étape la fragmentation, et en deuxième étape l'allocation de ces fragments aux sites.

L'approche top down est intéressante quand on part du néant. Si les BDs existent déjà la méthode bottom up est utilisée.

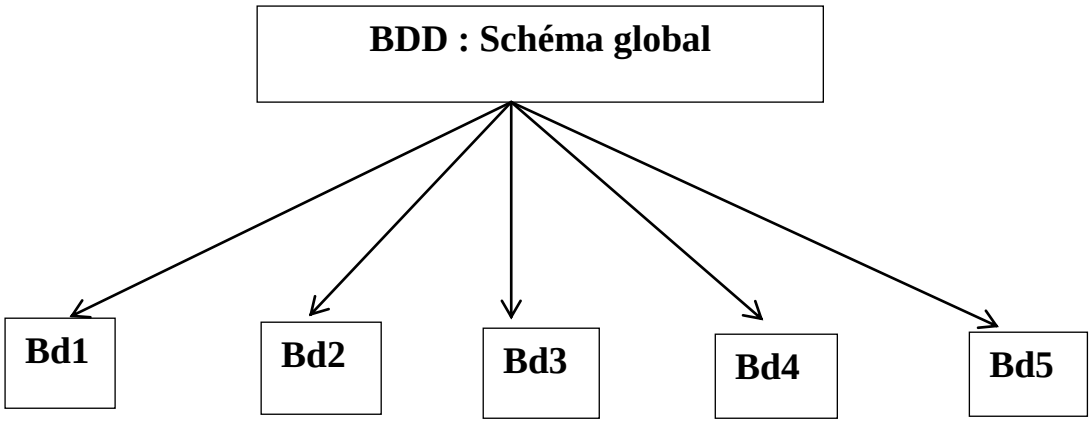


FIG.III.2. Architecture de la conception réparti descendante

Le principe de cette approche est :

- Conception du schéma conceptuel global.
- Distribution pour obtenir des schémas conceptuels locaux.
- Fragmentation.
- Affectation aux sites – Allocation.

- Environnement homogène
- Conception à partir de zéro
- Nouvelles étapes avant la conception physique
- Localisation des données
- Schémas locaux
- La distribution des données est bien présente
- Les tables du schéma global sont fragmentées (processus de fragmentation)
- Fragment : Sous-table obtenue par sélection de lignes et de colonnes à partir d'une table globale
- Les fragments sont donc placés sur des sites (processus d'allocation)

B. Conception ascendante (bottom up design)

L'approche se base sur le fait que la répartition est déjà faite, mais il faut réussir à intégrer les différentes BDs existantes en une seule BD globale. En d'autres termes, les schémas conceptuels locaux existent et il faut réussir à les unifier dans un schéma conceptuel global.

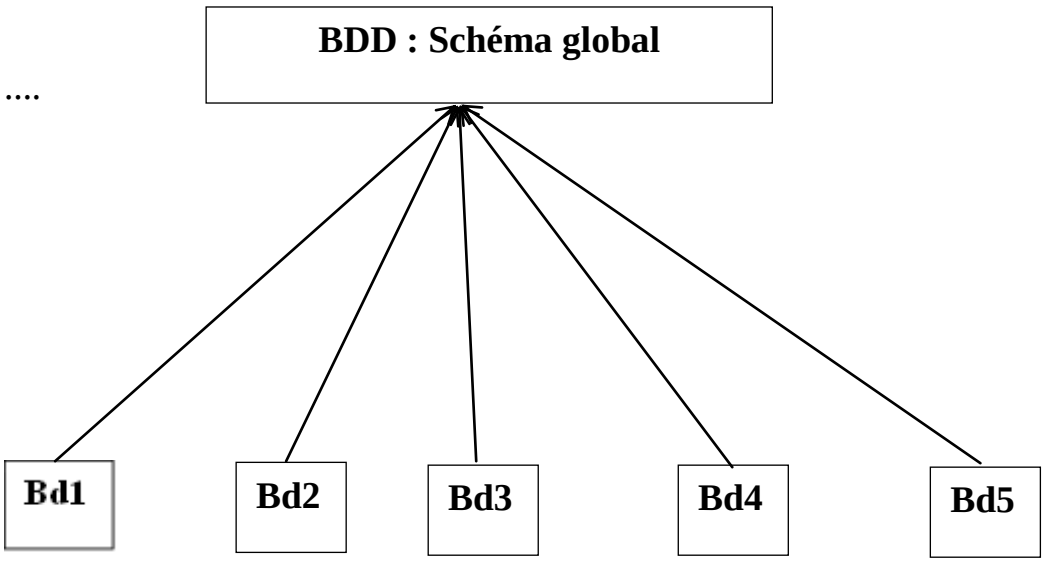
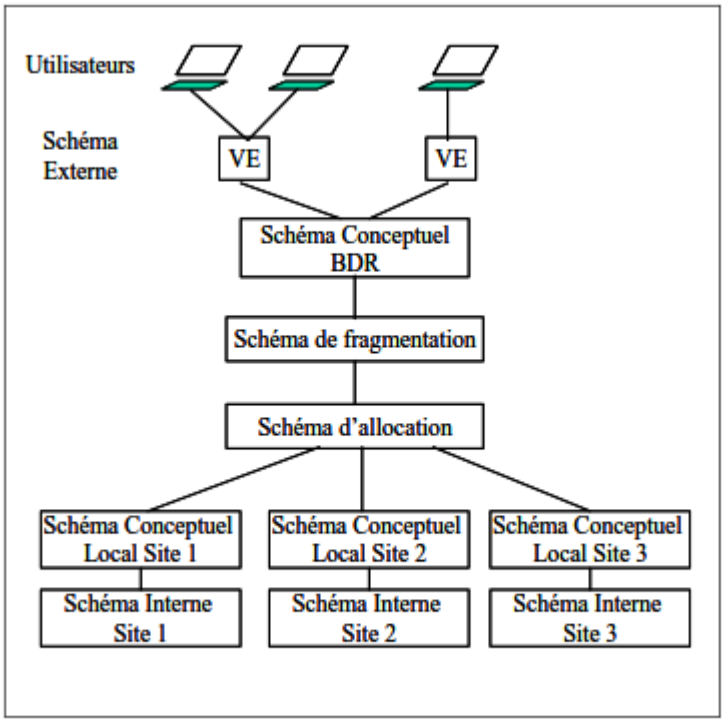


FIG.III.3. Architecture de la conception réparti ascendante

Le principe de cette approche est :

- Intégration de bases de données existantes.
- Hétérogénéité.
- On part de BD existantes (souvent hétérogènes)
- Intégration des BD locales existantes dans une seule base
- La distribution des données est préexistante
- Sémantique des schémas participants.



Architecture d'une BDR.

FIG.III.4. Architecture d'une base de données répartie

La répartition d'une base de données intervient dans les trois niveaux de son architecture en plus de la répartition physique des données :

- **Niveau externe:** les vues sont distribuées sur les sites utilisateurs.
- **Niveau conceptuel:** le schéma conceptuel des données est associé, par l'intermédiaire du schéma de répartition (lui-même décomposé en

un schéma de fragmentation et un schéma d'allocation), aux schémas locaux qui sont réparties sur plusieurs sites, les sites physiques.

- **Niveau interne:** le schéma interne global n'a pas d'existence réelle mais fait place à des schémas internes locaux répartis sur différents sites.

11.Fragmentation

La fragmentation est le processus de décomposition d'une base de donnée en un ensemble de sous-bases de données. Cette décomposition doit être sans perte d'information.

La fragmentation peut être coûteuse s'il existe des applications qui possèdent des besoins opposés.

Les règles de fragmentation sont les suivantes :

La complétude : pour toute donnée d'une relation R, il existe un fragment Ri de la relation R qui possède cette donnée.

La reconstruction : pour toute relation décomposée en un ensemble de fragments Ri, il existe une opération de reconstruction.

A. Objectifs de la décomposition

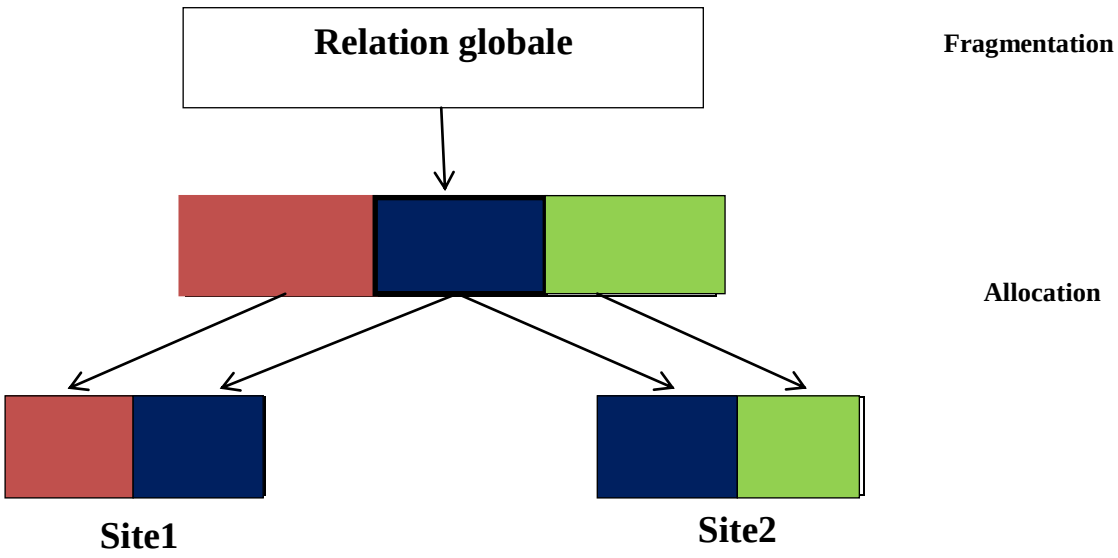


FIG.II.5. Fragmentation / Allocation

B. La réplication

B.1. Avantages

- Accès simplifié, plus performant pour les lectures
- Résistance aux pannes
- Parallélisme accru
- Evite des transferts

B.2. Inconvénients

- Overhead en mise à jour
- Cohérence des données

C. Pour quoi fragmenter

- Pas de vraie raison d'un point de vue « distribution de données ».
- Avantages de la distribution : performances, disponibilité, tolérance aux pannes, localité...
- Grain : une relation entière est une unité de distribution trop grande.

D. Comment fragmenter ?

- Grain / degré de fragmentation
  - Trop peu de fragments - faible concurrence.
  - Trop de fragments - surcoût dans la reconstruction des relations.
- Possibilités de fragmentation d'une relation

Exemple : Horizontale - basée sur des sélections

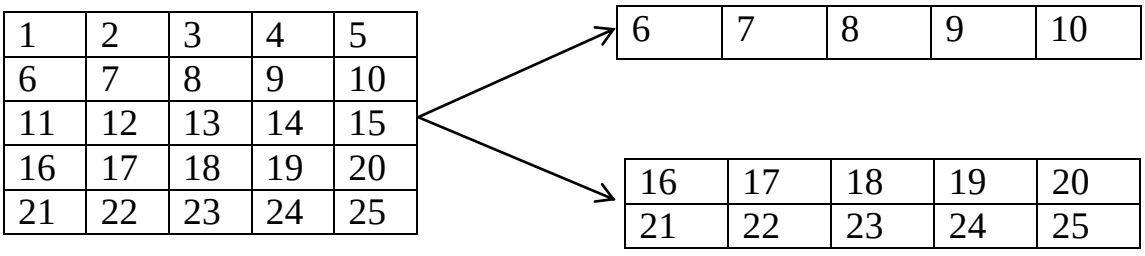


FIG.III.6. Fragmentation Horizontale

Exemple : Verticale - basée sur des projections.

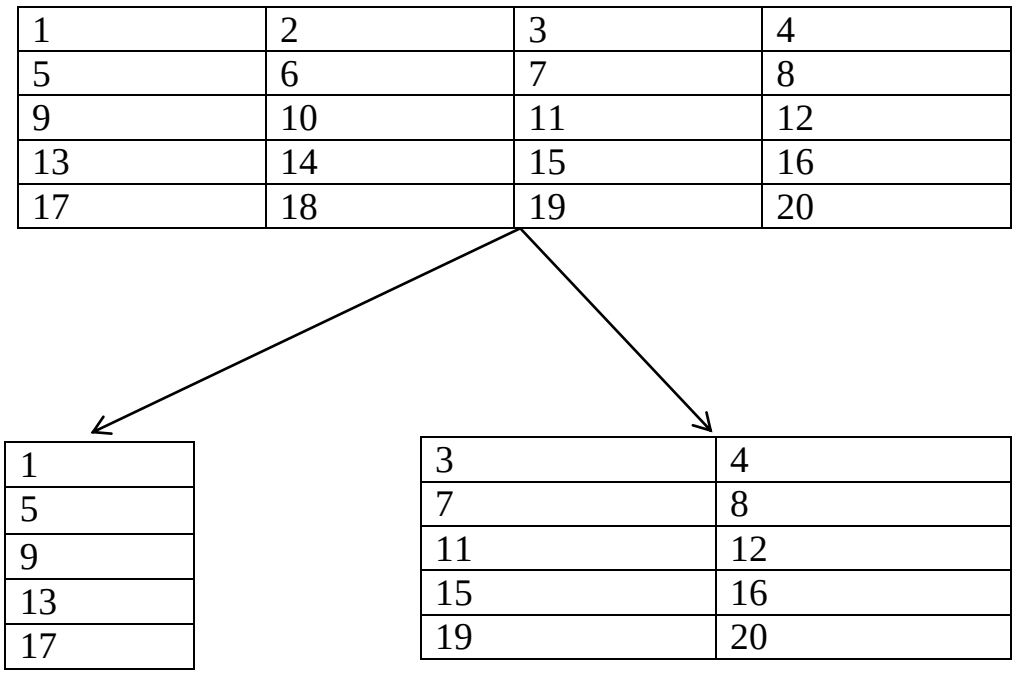


FIG.III.7. Fragmentation Verticale

Exemple : Possibilités de fragmentation d'une relation

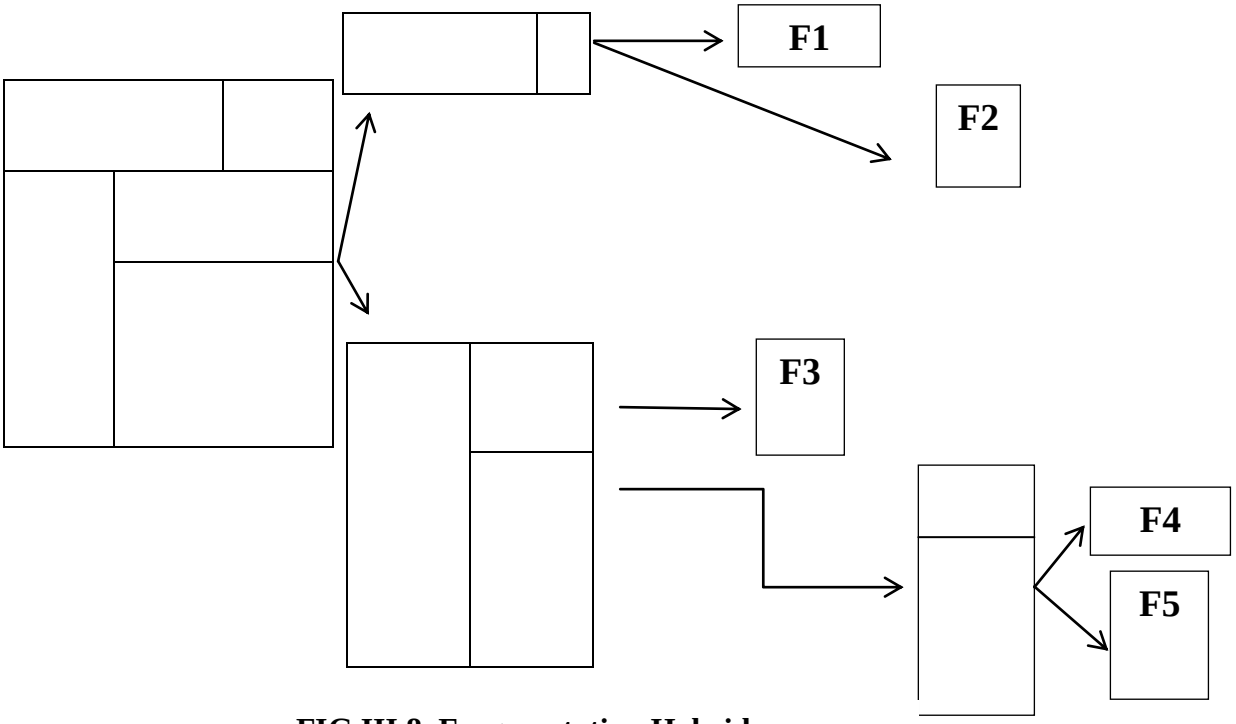


FIG.III.8. Fragmentation Hybride

E. Techniques de Fragmentation

Il existe plusieurs techniques de fragmentation, définies par l'unité de fragmentation.

A. Répartition des classes d'objet

Cette technique consiste en la répartition de classes (relation en relationnel, classe en Orienté-objet) qui peuvent être réparties sur différents fragments. Toutes les occurrences d'une même classe appartiennent ainsi au même fragment.

- L'opération de partitionnement est la définition de sous-schémas.
- L'opération de recomposition est la réunion de sous-schémas.

Dans l'exemple suivant la base de données relationnelle peut être fragmentée en **{Compte, Client} et {Agence}**

Relation Compte	No _client	Agence	Type compte	Somme
	174723	BNA412	courant	123345
	177498	BNA411	courant	34564
	201639	BNA410	courant	45102
	201639	BNA412	dépôt	325100
	203446	BNA410	courant	274882

Relation Agence	Agence	Adresse
	BNA412	Centre_Ville
	BNA411	Debedaba
	BNA410	Abdala
	BNA412	Centre_Ville
	BNA410	Abdala

Relation Client	No_ client	Non_client	Prénon_client	Age
	174723	Seddiki	Noureddine	29
	177498	Benahmed	Khelifa	38
	201639	Belgachi	Mohammed	51
	203446	Benfilali	Mostefa	36

1. Répartition des occurrences (fragmentation horizontale)

Les occurrences d'une même classe peuvent être réparties dans des fragments différents.

- L'opérateur de partitionnement est la sélection (σ)
- L'opérateur de recomposition est l'union (∪)

Dans l'exemple précédent, la relation Compte peut être fractionnée en Compte1 et Compte2 avec la fragmentation suivante :

Compte1 = $\sigma_{[TypeCompte = 'courant']}$ Compte

et Compte2 = $\sigma_{[TypeCompte = 'dépôt']}$ Compte

La recomposition de Compte est Compte1 ∪ Compte2

B. Répartition des attributs (fragmentation verticale)

Toutes les valeurs des occurrences pour un même attribut se trouvent dans le même fragment. Une fragmentation verticale est utile pour distribuer les parties des données sur le site où chacune de ces parties est utilisée.

- L'opérateur de partitionnement est la projection (π)
- L'opérateur de recomposition est la jointure

Soit le partitionnement de la relation précédente Client en deux relations :

Cli1 = $\pi_{[NoClient, NomClient]}$ Client

et Cli2 = $\pi_{[Noclient, Prénom, Age]}$ Client

No_Client	Nom_Client
174 723	Seddiki
177 498	Belgachi
201 639	Benahmed
203 446	Benfilali

No_Client	Préno, _Client	Age
174 723	Noureddine	29
177 498	Mohammed	38
201 639	Khekifa	51
203 446	Mostefa	36

La relation d'origine est obtenue avec la recomposition suivante : **Client = Cli1 * Cli2**

a. Répartition des valeurs (fragmentation hybride)

C'est la combinaison des deux fragmentations précédentes, horizontale et verticale.

Les occurrences et les attributs peuvent donc être répartis dans des partitions différentes.

- L'opération de partitionnement est une combinaison de projections et de sélections.
- L'opération de recomposition est une combinaison de jointures et d'unions.

La relation Client est obtenue avec : (Cli3 ∪ Cli5) * Cli4 * Cli6

Relation Cli3 $\pi_{[NoClient, NomClient]}$ ($\sigma_{[Age < 38]}$Client)

Relation Cli5 $\pi_{[NoClient, NomClient]}$ ($\sigma_{[Age \geq 38]}$Client)

Relation Cli4 $\pi_{[No_Client, Prénom]}$Client

Relation Cli6 $\pi_{[NoClient, Age]}$Client

12.Schéma d'allocation

L'affectation des fragments sur les sites est décidée en fonction de l'origine prévue des requêtes qui ont servi à la fragmentation. Le but est de placer les fragments sur les sites où ils sont le plus utilisés, et ce pour minimiser les transferts de données entre les sites.

L'allocation peut se faire avec réplication ou sans réplication. Sachant que la réplication favorise les performances des requêtes et la disponibilité des données, mais elle est coûteuse en considérant les mises à jour des fragments répliquas.

13.Des exercices

Exercice 01 :

Objectif : Fragmenter la relation Compte (No_Client, Agence, Type_Compte, Somme).

Proposer un schéma de fragmentation horizontale, puis verticale en tenant compte des requêtes suivantes :

R1 = $\pi_{[NoClient, Agence]}$ ($\sigma_{[(TypeCompte = 'courant') \wedge (Somme > 100\ 000)]}$ Compte)

R2 = $\sigma_{[Agence = 'Lausanne']}$ Compte

R3 = $\pi_{[NoClient, Somme]}$ ($\sigma_{[(Agence = 'Genève') \wedge (TypeCompte = 'courant')]}$ Compte)

Exercice 02 :

Rémunération (Titre, salaire)

Employé (num_E, nom_E, Titre)

Projet (num_P, nom_P, budget, ville)

Affectation (num_E, num_P, responsabilité, durée)

1. Proposer un schéma de fragmentation horizontale pour la relation Rémunération.
2. A partir des requêtes R1 et R2, proposer un schéma de fragmentation horizontale pour la relation Projet.

R1 : SELECT nom_P, budget FROM Projet WHERE **ville = valeur** ;

R2 : SELECT * FROM projet WHERE budget < 200 000 ;

3. Fragmenter Employé selon les fragments de Rémunération.
4. Quels sont les choix de fragmentation de Affectation.

14.Conclusion

Une base de données répartie est une collection de données logiquement unies et physiquement réparties sur plusieurs machines interconnectées par un réseau de communication. Elle permet d'intégrer et de partager des données gérées par des systèmes de gestion des bases de données réparties.

Dans ce chapitre, nous avons présenté les bases de données répartir, cette technique à été inventée pour faciliter l'évolution d'applications répartir.

Chapitre IV
Les bases de données Mulimudias

IV. Les bases de données multimédias

1. Introduction

Les bases de données en générale est un ensemble structuré et organisé permettant le stockage de grandes quantités d'informations afin d'en faciliter l'exploitation (ajout, mise à jour, recherche de données) pour des raisons :

- Qui utilise plusieurs moyens de diffusion?
- Qui concerne plusieurs média?
- Diffusé par plusieurs média?
- Qui utilise ou concerne plusieurs médias?

Comme son nom ne l'indique pas le multimédia se concentre généralement sur un support unique (MONOmedia). C'est le contenu du support qui est multiple et riche : simple texte, images, sons, vidéo.
Cette richesse et cette variété font que l'intégration, et le respect de standards sont des enjeux importants. Le contenu multimédia peut être stocké et accédé de différentes manières :

- faible volume (qq centaines de MO) circonscrit sur un CD/DVD ROM et accessible via une IHM spécifique.
- volume important (GO ou TO) dans une BD multimédia et accédé via un langage de requête spécifique.
- volume illimité sur le WEB accédé de manière associative via la navigation hypermédia.

A la différence du textuel, ces nouveaux contenus induisent une forte interactivité avec l'utilisateur.

2. Problématique

On le voit le problème principal vient de l'indexation de leur contenu et des recherches par contenu.

- Connaître les données que l'on traite(Acquisition).
- Connaître les manières de les structurer (Stockage).

- Connaître les manières de les utiliser (Utilisation).

A. Volume des données :

- **Quelques chiffres**…
  - 100 lignes de fax : 6,4 MO
  - 100 images couleur 500 MO
  - 1 heure de vidéo : 1 GO
- **Stockage**
  - Compression indispensable, mais cela a un coût (temps).
  - Hiérarchies de mémoires, plusieurs supports (complique le fonctionnement du SGBD).
- **Efficacité de la recherche**
  - Index adaptés.

1. Types de données

- **Texte**
  - Les modèles classiques ne permettent pas le traitement efficace.
  - Documents fortement structurés.
- **Image**
  - Nombreuses applications.
  - Description externe + description du contenu.
  - Recherches sur le contenu.
  - Différents codages, différentes normes.
  - Visualisation des résultats.
- **Sons**
  - Problème de recherche et traitement
- **Vidéo**
  - Séquences d'images fixes ?
  - Interrogation des vidéos
  - Opérations de type magnétoscope

3. Définitions

A. Media: Tout support de diffusion de l'information constituant à la fois un moyen d'expression et un intermédiaire transmettant un message à l'attention d'un groupe.

B. Le mot multimédia : est apparu vers la fin des années 1980, lorsque les CD-ROM se sont développés. Il désignait alors les applications qui, grâce à la mémoire du CD et aux capacités de l'ordinateur, pouvaient générer, utiliser ou piloter différents médias simultanément. Aujourd'hui on utilise le mot multimédia pour désigner toute application utilisant ou servant à travailler sur au moins un média spécifique. Par ailleurs, en recherche en informatique, on nomme multimédia l'étude des médias non textuels, principalement les images, les vidéo et les sons.

Aujourd'hui on utilise le mot multimédia pour désigner toute application utilisant ou servant à travailler sur au moins un média spécifique.
Par ailleurs, en recherche en informatique, on nomme multimédia l'étude des médias non textuels, principalement les images, les vidéo et les sons.

C. **Une base de données multimédia :** est un type de base de données consacré au stockage et à l'organisation de données multimédia : documents sonores, images, vidéos.
Leur apparition et leur développement récents sont une tentative de réponse non seulement à un besoin de stockage de tels documents, mais également à un besoin de possibilités accrues pour leur traitement.
Elles peuvent s'appuyer sur différentes architectures de bases de données, les types les plus utilisés étant le modèle relationnel et le modèle objet.

D. **Les premières versions de BDMM :** étaient simplement de la présentation : afficher une photo des employés en plus des informations structurées classiques, afficher une image du produit pour enrichir le catalogue.Le stockage externe de ces images s'avérait suffisant et la BD servait à manipuler les données connexes.
Ceci est toujours d'actualité, mais constitue une 'version pauvre' des BDMMs.

Au de la des fonctionnalités clssiques d'un SGBD, qui sont également utiles à une BDMM :

- gestion de gros volumes,
- langage de requete et de mise à jour,
- sécurité (intégrité, disponibilité, fiabilité).
- gestion des accès conurrents,
- performance

4. Types de données

A. Texte

- Les modèles classiques ne permettent pas le traitement efficace de documents fortement structurés.

B. Image

- Nombreuses applications
- Description externe + description du contenu
- Recherches sur le contenu

- Différents codages, différentes normes
- Visualisation des résultats

C. Sons

- Problème de recherche et traitement

D. Vidéo

- Séquences d'images fixes ?
- Interrogation des vidéos
- Opérations de type magnétoscope

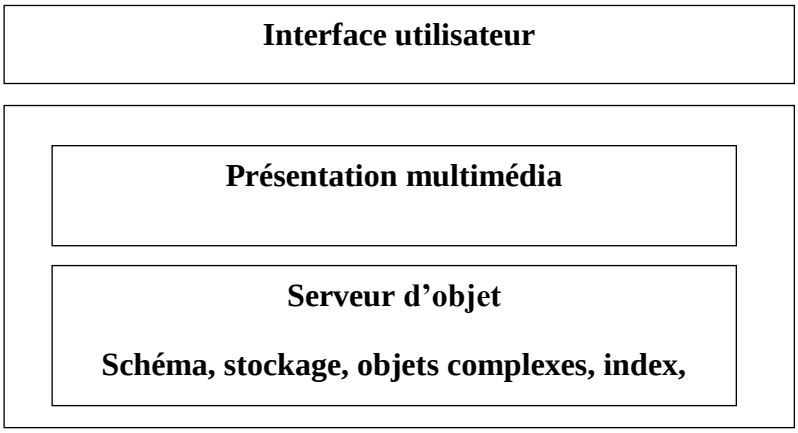


FIG.IV.1 Architecture fonctionnelle d'une BDM

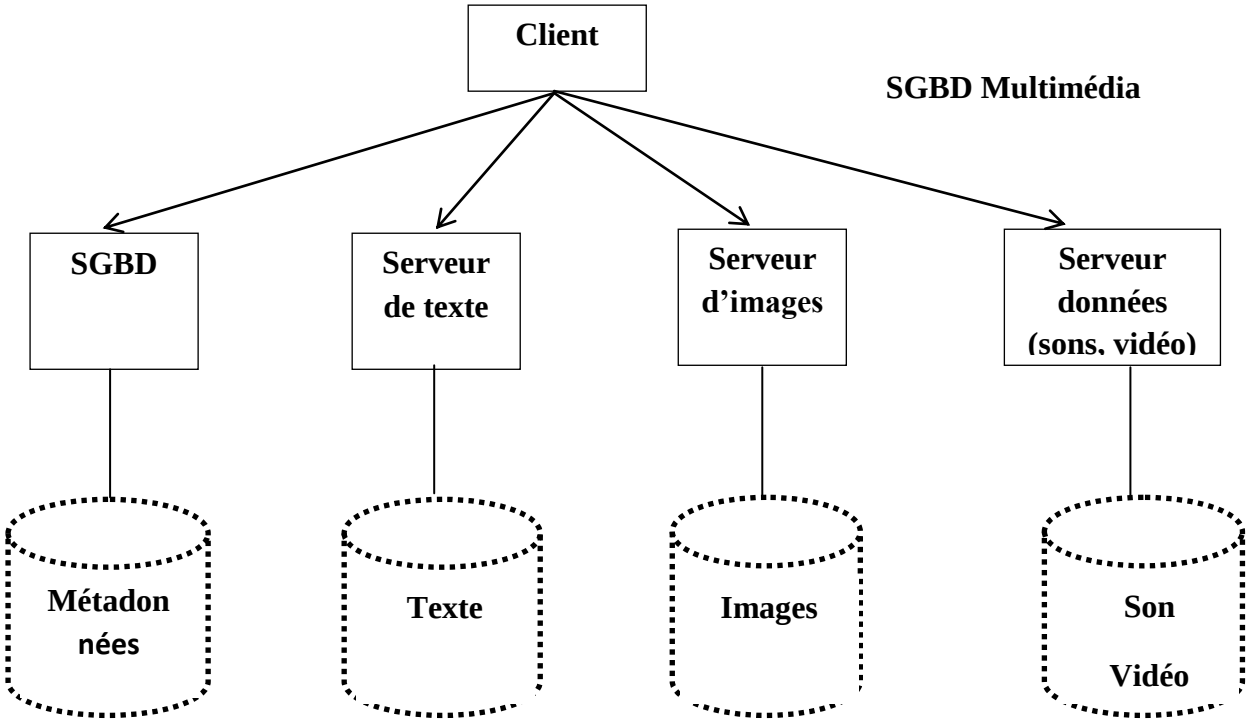


FIG. IV.2 Architecture physique d'une BDM

5. Type de base de données multimédia

Parmi les types de base de données on trouve :

A. Les bases de données génériques :

- Contenu hétérogène (bases grand public, Internet, archives généralistes).
- En général sans vérité-terrain naturelle.

B. Les bases de données spécifiques :

- Spécifiques à un domaine d'application

6. Caractéristiques

Une base multimédia à la particularité de représenter les documents intégralement, c'est-à-dire qu'elle ne se satisfait pas de stocker une référence vers des documents, mais qu'elle intègre leur contenu.

De plus, un SGBD multimédia doit permettre la modélisation d'objets réels mais également de leurs interactions. Il ne s'agit donc pas seulement de la définition des

liens objet à objet grâce à des métadonnées, mais réellement d'une représentation des objets et de leurs liens grâce à une modélisation orientée objet.

Les documents peuvent être classés suivant trois types :

1. **Statiques** : média dont l'état n'évolue pas durant sa lecture, à l'instar des textes et images.
2. **Dynamiques :** média qui possède un contenu dont l'état lu évolue au fil du temps. Le son et la vidéo sont les exemples les plus communs.
3. **Dimensionnels :** objets 3D.

Medium	Éléments	Dépendance au temps
Texte	caractères	Statique
Graphique	Vecteurs, régions	Statique
Image	Pixels	Statique
Audio	Son, volume	Dynamique
Vidéo	Images raster, graphiques	Dynamique

Tab IV.1 caractéristique de type de données

7. Domaines d'application

Parmi les domaines d'application :

A. Applications scientifiques

- Bases d'images médicales
    - Exemple : retrouver les images présentant un caractère pathologique, dans un but éducatif ou diagnostic.
- Bases d'images botaniques
    - Exemple : identifier les gènes qui interviennent dans une même chaîne de synthèse protéinique.

B. Bases d'images satellitaires

- Audiovisuel
    - Exemple : retrouver un plan spécifique d'un film ou d'un journal télévisé (TV, INA), annotation automatique de vidéos, détection de copies (droits).

C. Authentification

- Exemple : détecter les contrefaçons de modèles déposés, identifier un visage (Visionics) ou des empreintes digitales (Thomson Idmatics), supports d'enquête (Police Judiciaire).

D. Le design, la publicité, l'architecture

- Exemple : rechercher une texture spécifique pour l'industrie textile, illustrer une publicité par une photo adéquate, mettre en lumière des tendances.

E. Internet

- Exemple : butinage web (Ibazar.com), commerce électronique.
- moteur de recherche, vidéo à la demande, stockage en ligne…

F. Armée et Sécurité

- Exemple : (Imagerie satellite, enregistrement audio…)

G. Journalisme

- Exemple : (Bibliothèque numérique, documents de type variés)

8. Pourquoi les bases de données multimédia ?

A. De l'importance du Multimédia

On observe ces dernières années :

- une augmentation vertigineuse du nombre des contenus digitaux.
    - diversification des contenus : image statique, gif animé,flash, MPEG, DIVx, stream video, mais aussi MP3, stream audio, etc.)
    - Meilleure disponibilité des contenus : moteurs de recherche spécialisés, peer to peer
- une sensible baisse des coûts alliée à une augmentation de performances.
    - Des périphériques de stockage (mémoire vive et disque), des CPUs.
    - Démocratisation des outils de production et de restitution digitaux (scanner, appareil photo/video, téléphone portable, lecteurs/encodeur de poche, etc.)
- qu'Internet commence enfin à prendre sa vraie dimension Hypermedia et plus seulement hypertexte.

B. De l'importance des Bases de données

Corolairement l'inflation d'information non structurée, au départ textuelle et désormais multimédia, a imposé l'utilisation complémentaire de systèmes de gestion d'informations structurés, pour pallier les limites inhérentes à l'hyper navigation (il n'est que de constater le succès d'outils comme Google ou Yahoo). Les bases de données sont intensivement utilisées :

- Pour produire des pages dynamiques
- Pour rechercher efficacement une information pertinente

9. Système multimédia

Une base de données multimédia s'intègre dans un système multimédia plus général et qui comprend au moins 4 composants :

- la BDMM elle-même,
- Un serveur de stockage,
- Un réseau,
- Des clients MM fixes ou non.

Les principaux processus d'un système multimédia sont :

- La capture des données,
- Le stockage,
- L'archivage,
- La recherche et la fourniture d'informations.

10. Système de gestion de base de données multimédia

Un système de gestion de bases de données multimédia doit :

- Offrir les mêmes caractéristiques qu'un SGBD classique (persistance, gestion des transactions, interrogation …)
- Permettre de représenter, interroger et manipuler tout type de media.

Les recherches actuelles : représentation, stockage, interrogation, liaison image et sémantique Intégration du multimédia dans les BD classiques

Exemple : package (ORDSYS) de classes fourni par Oracle (8 – objet-relationnel) pour gérer des données multimédia :

- **localData :** contient les données multimédia stockées localement comme un BLOB (Binary Large Object – jusqu'à 4 GB).
- **srcType** : identifie le type de source des données (FILE, http, …)
- **srcLocation :** identifie où les données peuvent être trouvées (dépendant de srcType (FILE -> directory, HTTP -> URL ou user-defined).
- **srcName :** identifie le nom des données.
- **updateTime :** la date à laquelle le fichier a été modifié pour la dernière fois.
- **local :** valeur qui détermine si le fichier est local ou pas (1 = BLOB, 2 = externe).

11. Conclusion

Dans ce chapitre, comme son nom ne l'indique pas le multimedia se concentre généralement sur un support unique (MONOmedia). C'est le contenu du support qui est multiple et riche : simple texte, images, sons, vidéo.
Cette richesse et cette variété font que l'intégration, et le respect de standards sont des enjeux importants.

Le contenu multimedia peut être stocké et accédé de différentes manières :

- faible volume (qq centaines de MO) circonscrit sur un **CD/DVD** ROM et accessible via une IHM spécifique
- volume important (GO ou TO) dans une **BD multimédia** et accédé via un langage de requête spécifique
- volume illimité sur le **WEB** accédé de manière associative via la navigation hypermédia

Chapitre V
Les bases de données Déductives

V. Les Bases de Déductive

1. Introduction

Depuis que la notion de base de données déductive est bien comprise, sous la pression des applications potentielles, les concepteurs de systèmes s'efforcent de proposer des algorithmes et méthodes efficaces pour réaliser des SGBD déductifs traitant de grands volumes de faits (les bases de données classiques) et de règles. L'objectif est de fournir un outil performant pour aider à la résolution de problèmes, exprimés sous forme de requêtes, dont la solution nécessite des volumes importants de données et de règles. Les applications potentielles sont nombreuses. Outre la gestion classique ou prévisionnelle, nous citerons par exemple l'aide à la décision, la médecine, la robotique, la productique et plus généralement toutes les applications de type systèmes experts nécessitant de grands volumes de données. Il a même été possible de penser que l'écriture de règles référençant de grandes bases de données remplacerait la programmation classique, au moins pour les applications de gestion. Ces espoirs ont été quelque peu déçus, au moins jusqu'à aujourd'hui.

Ce chapitre introduit la problématique des SGBD déductifs, puis présente le langage standard d'expression de règles portant sur de grandes bases de données, appelé DATALOG.

Les questions qui se posent sont les suivantes :

- Quelles est la forme des règles que l'on souhaite utiliser ? Comment les interpréter ?
- Les requêtes répressibles en SQL peuvent-elles être exprimées par des règles ?
- Les règles peuvent-elles exprimer plus ?

2. Problématique des SGBD Déductifs

Un SGBD déductif est tout d'abord un SGBD. En ce sens, il doit posséder un langage de description de données permettant de définir les structures des prédicats de la base B1, B2, ... Bn, par exemple sous forme de relations, et les contraintes d'intégrité associées. Il offre aussi un langage de requête permettant de poser des questions et d'effectuer des mises à jour. Ces deux langages peuvent être intégrés et posséder une syntaxe propre, ou plusieurs, offertes aux usagers. Parmi ces langages, il est permis de penser que SQL restera une des interfaces offertes par un SGBD déductif, surtout devant la poussée de sa normalisation.

A. Architecture type d'un SGBD déductif intégré

Le SGBD est dit déductif car il permet de déduire des informations à partir de données stockées par utilisation d'un mécanisme d'inférence logique. Les informations sont les tuples des prédicats intentionnels ; elles peuvent être déduites lors de l'interrogation des prédicats intentionnels ou lors des mises à jour des prédicats extensionnels. La mise à jour des prédicats intentionnelles est difficile : il faut théoriquement répercuter sur les prédicats extensionnels, ce qui nécessite une extension des mécanismes de mise à jour au travers de vues.

Un SGBD permettant de dériver les tuples de prédicats intentionnels par utilisation de règles.

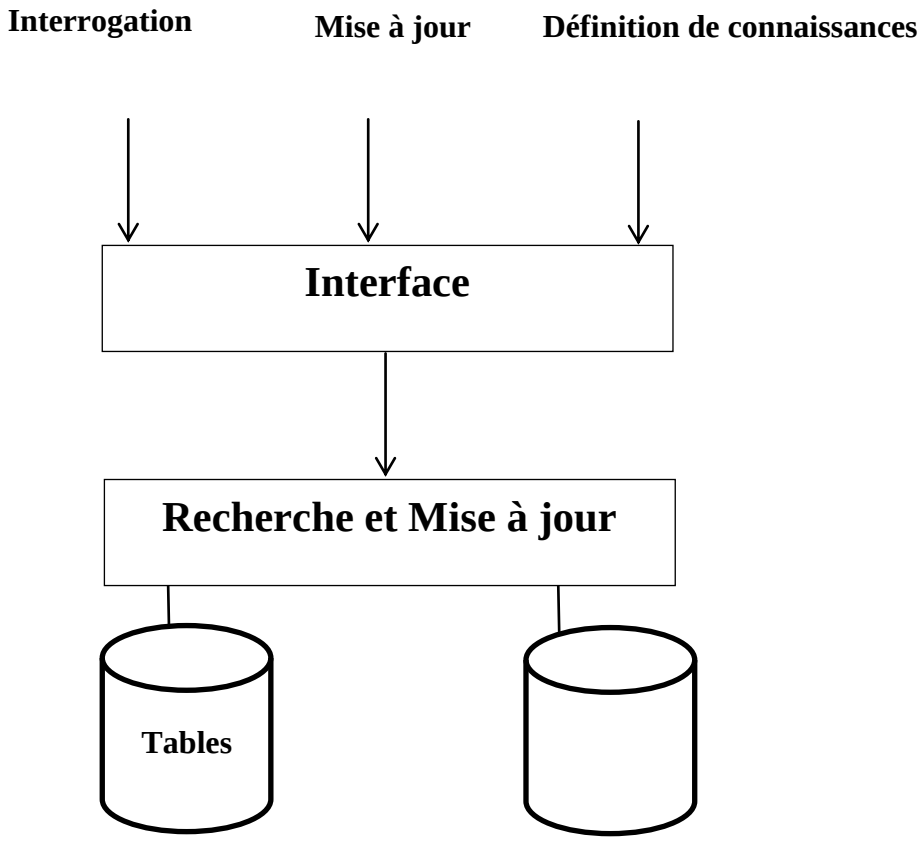


FIG.V.1. Architecture d'un SGBD déductif

3. Définition

Une base de données déductive (BDD) est constituée :

- D'un ensemble de prédicats (relations), dits de base ou extensionnels, dont l'extension est conservée explicitement dans la base de données :
  - la base de données extensionnelle
- d'un ensemble de prédicats (relations), dits dérivés ou intentionnels, dont l'extension est définie par des règles déductives:
  - la base de données intentionnelle

Exemple

Base de données **extensionnelle** : Prédicat parent `a 2 arguments (ou

relation parent `a 2 attributs) :

Parent	
Mohammed	Amina
Ali	Ahmed
Malika	Farid
Fatima	Fatiha
……..	…….

4. Type de base de données déductif

a. Base extensionnelle

Dans le contexte logique, une base de données est perçue comme un ensemble de prédicats. Les extensions des prédicats extensionnels sont matérialisées dans la base de données. Les prédicats extensionnels correspondent aux relations du modèle relationnel.

Prédicat extensionnel c'est un prédicat dont les instances sont stockées dans la base de données sous forme de tuples :

- Correspond à l'ensemble des faits disponibles
- Est constitué du contenu de la base
- En BD relationnelle: l'ensemble des tuples des relations

Base de données **intentionnelle** : Prédicat grand parent `a 2 arguments

(ou relation grand parent `a 2 attributs)

grandparent(X,Y) ← parent(X,Z) AND parent (Z,Y)

B. Base intentionnelle

Une base de données est manipulée par des programmes logiques constitués d'une suite de clauses de Horn qui définissent des prédicats intentionnels. Un prédicat intentionnel est donc défini par un programme de règles logiques ; il correspond à une vue du modèle relationnel.

Prédicat intentionnel c'est un prédicat calculé par un programme constitué de règles logiques dont les instances ne sont pas stockées dans la base de données :

- Correspond aux faits qui peuvent être déduits
- Les données ne sont pas présentes dans la base
- Des règles définissent le moyen de générer ces nouveaux faits

Une base de données logique est constituée d'un ensemble de prédicats extensionnels constituant la base de données extensionnelle et d'un ensemble de prédicats intentionnels constituant la base de données intentionnelle. Les règles permettant de calculer les instances des prédicats intentionnels sont donc partie intégrante de la base de données logique. Elles sont écrites dans le langage DATALOG basé sur les clauses de Horn. La **FIG.V.2** illustre les notions de bases de données extensionnelle et intentionnelle, la seconde étant dérivée de la première par des règles stockées dans la méta-base du SGBD.

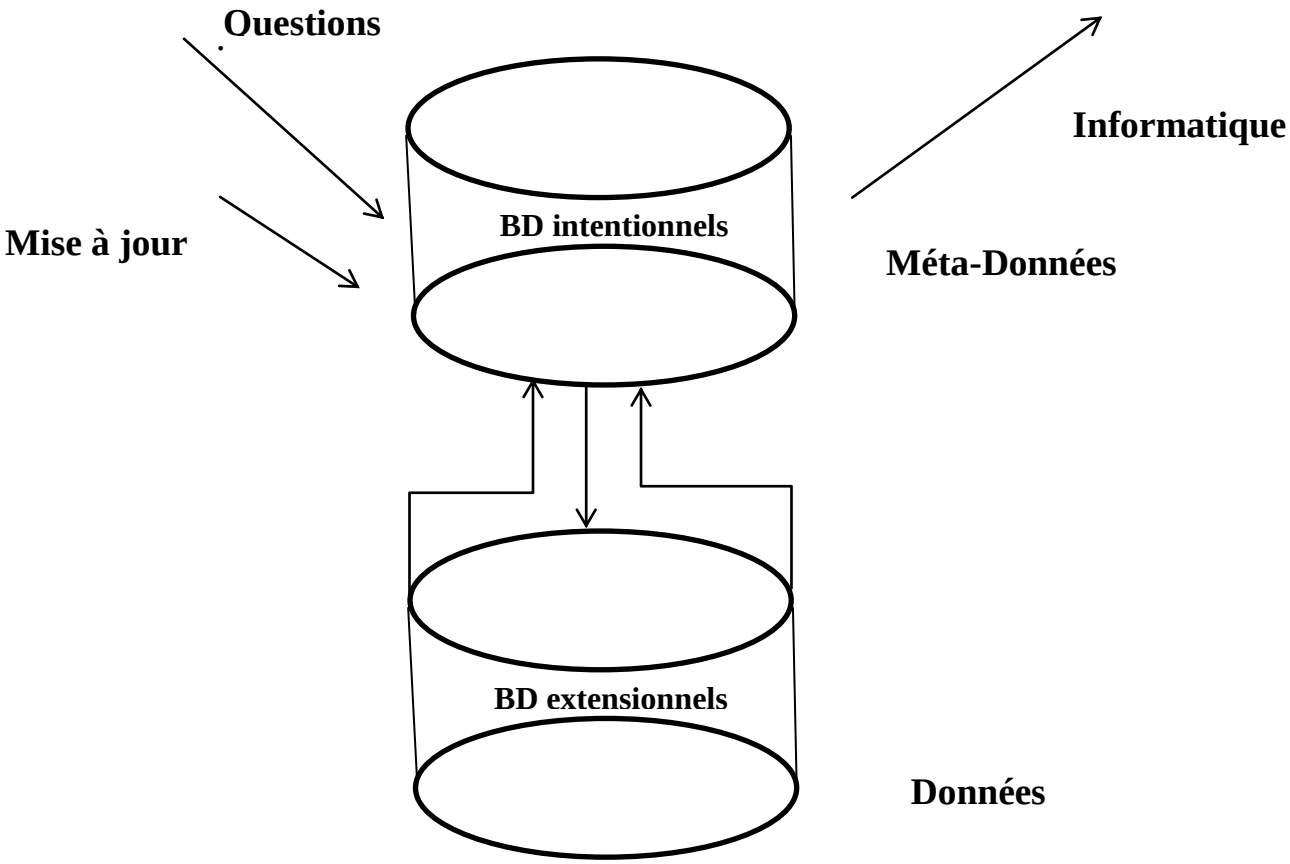


FIG.V.2. Base de données extensionnelle et intentionnelle

5. Le langage DATALOG

Le langage DATALOG est dérivé de la logique du premier ordre (PROLOG). C'est à la fois un langage de description et de manipulation de données. Le modèle de description de données supporté par DATALOG est essentiellement relationnel, une relation étant vue comme un prédicat de la logique. Le langage de manipulation est un langage de règles bâti à partir des clauses de Horn. Le nom DATALOG signifie « logique pour les données ». Il a été inventé pour suggérer une version de PROLOG (le langage de programmation logique) utilisable pour les données.

A. Description de langage DATALOG

- Langage logique
- Inspiré de Prolog
- Destiné aux bases de données
- Langage de bases de données
- DATA et Logique
- Manipulation de données à l'aide de la logique
- Puissance d'expression plus grande que SQL

- Langage pour les BD déductives
- À la base des prototypes
- Permet de comparer la puissance d'expression des autres langages

B. Syntaxe de DATALOG

Outre la définition des prédicats extensionnels, DATALOG permet d'écrire les clauses de Horn spécifiant les prédicats intentionnels. L'alphabet utilisé est directement dérivé de celui de la logique du premier ordre. Il est composé des symboles suivants :

- Des **variables** dénotées x, y, z … ;
- Des **constantes** choisies parmi les instances des types de base entier, numérique et chaînes de caractères ;
- Des prédicats relationnels dénotés par une chaîne de caractères, chaque prédicat pouvant recevoir un nombre fixe d'arguments (n pour un prédicat n-aire) ;
- Les **prédicats de comparaison** =, <, >, <=, ...
- Les connecteurs logiques « et » dénoté par une virgule (,) et « implique » que l'on interprète de la droite vers la gauche, dénoté par le signe d'implication inversé (⟵).

6. Conclusion

Cette partie a proposé une synthèse des recherches et développements en matière de bases de données déductives. Parmi les multiples approches, nous avons surtout insisté sur l'intégration d'un langage de règles au sein d'un SGBD relationnel étendu.

Conclusion

Actuellement, la plupart des logiciels de bases de données (SGBD) reposent sur le modèle relationnel, réparti et orientée objet. Une base de données est vue comme un ensemble de tables ou des objets , structurées selon des formes normales et manipulées grâce à des langages non algorithmiques dont le représentant le plus connu est SQL.

Les bases de données avancées permettent la réalisation de diverses applications, néanmoins, elles doivent rester cohérentes au fur et à mesure que ces données sont mises à jour.
La cohérence d'une base dépend de la qualité de son schéma et de la mise en place des contraintes.
La vérification des contraintes dans une base de données est efficace si elle profite des mécanismes (déclaratifs) que les SGBD avancées tell que SBBDO, SGBDR, SGBDM et SGBDM laissent à notre disposition.

Références

1. Michel Crucianu & Valérie Gouet-Brunet - 4 janvier 2016 - Bases de données multimédia , Université Paris Dauphine.
2. Ladjel Bellatreche , 2009, Bases de Données Réparties, Univ.Poitiers France 2009.
3. Nicolas DURAND, 2008, Bases de données , Université de la Méditerranée - Aix-Marseille II France.
4. Gardarin, G, 1999, Bases de Données Objet et Relationnel, Eyrolles, France.
5. GRIN 2006] Modèle objet – relationnel SQL99 Richard Grin Université de Nice Sophia-Antipolis France.
6. KERHERVÉ UQAM Canada 2005, Intégrité et Bases de Données Actives Brigitte.
7. Brigitte KERHERVÉ UQAM Canada 2005.Bases de Données Déductives
8. Cours de Master 02 Filière Académique " Informatique Fondament Université de Mentouri de Constantine Algérie Décembre 2010.
9. Paton, N., Diaz, O., 1999, Active Database Systems.
10. M. T. Özsu & P. Valduriez, Principles of Distributed Database Systems, Prentice Hall 1999.
11. J. Durbin, L. Ashdown, Oracle8i : Distributed Database Systems, Oracle Press, 1999.
12. B. Ducourthial, Les bases de données réparties,