BD :
BASE DE DONNÉES

REZEG Khaled

Maître de conférences B

Dpt d'Informatique – Université de Biskra

B.P. 145 Biskra 07000

Tel & Fax : 033 74 31 61

Email : rezeg_khaled@yahoo.fr

BD:
FICHE D'IDENTIFICATION

Nom & prénom : ……

E-mail : ……

Comment vous sauvegardez vos données ?

Que représente les BD pour vous ?

Quels sont les langages de programmation que vous connaissez?

BASE DE DONNÉES
BD

Objectif:

- Comprendre les objectifs, les architectures et les langages de bases de données.
- Maîtriser les fondements théoriques et les algorithmes de base des systèmes de gestion de bases de données, depuis la conception de base de données jusqu'au traitement de requêtes et la gestion de transactions.
- Le module s'appuie sur le modèle relationnel et les langages associés, en particulier SQL.

Programme

BASE DE DONNÉES
BD

- **Généralités sur les bases de données**
- **Modélisation de bases de données : Le modèle relationnel**
- **Conception et optimisation de schéma relationnel**
- **Présentation générale de SQL**
- **Introduction à l'algèbre relationnelle**
- **Evaluation et optimisation de requête**
- **Transaction**

Programme

BASE DE DONNÉES
BD

- **Références**
- **Base de données à l'usage des étudiants Dr. Brahim BELATTAR - LISA - Dpt d'informatique - Faculté des sciences de l'Ingénieur - Univ. de Batna - 05000 – Algérie**
- Georges Gardarin. Bases de données: objet et relationnel. Eyrolles, 1999.
- **Base de données série des pages blues**

- **Mode d'évaluation.**
- 50 % Examen + 25 % TP + 25 % TD.

Chapitre 01

Généralités sur les bases de données

Généralités sur les bases de données

- Qu’est-ce donc qu’une base de données ?
- Que peut-on attendre d’un système de gestion de bases de données ?
- Que peut-on faire avec une base de données ?

Des données ? Est ce important pour vous ?

- Des relevés de banques, de cartes de crédit
- Des carnets d’adresses
- La consommation de téléphone
- Des inscriptions à des clubs, associations,
- Des papiers utiles
- Des horaires et disponibilités de transport
- Des programmes de télé

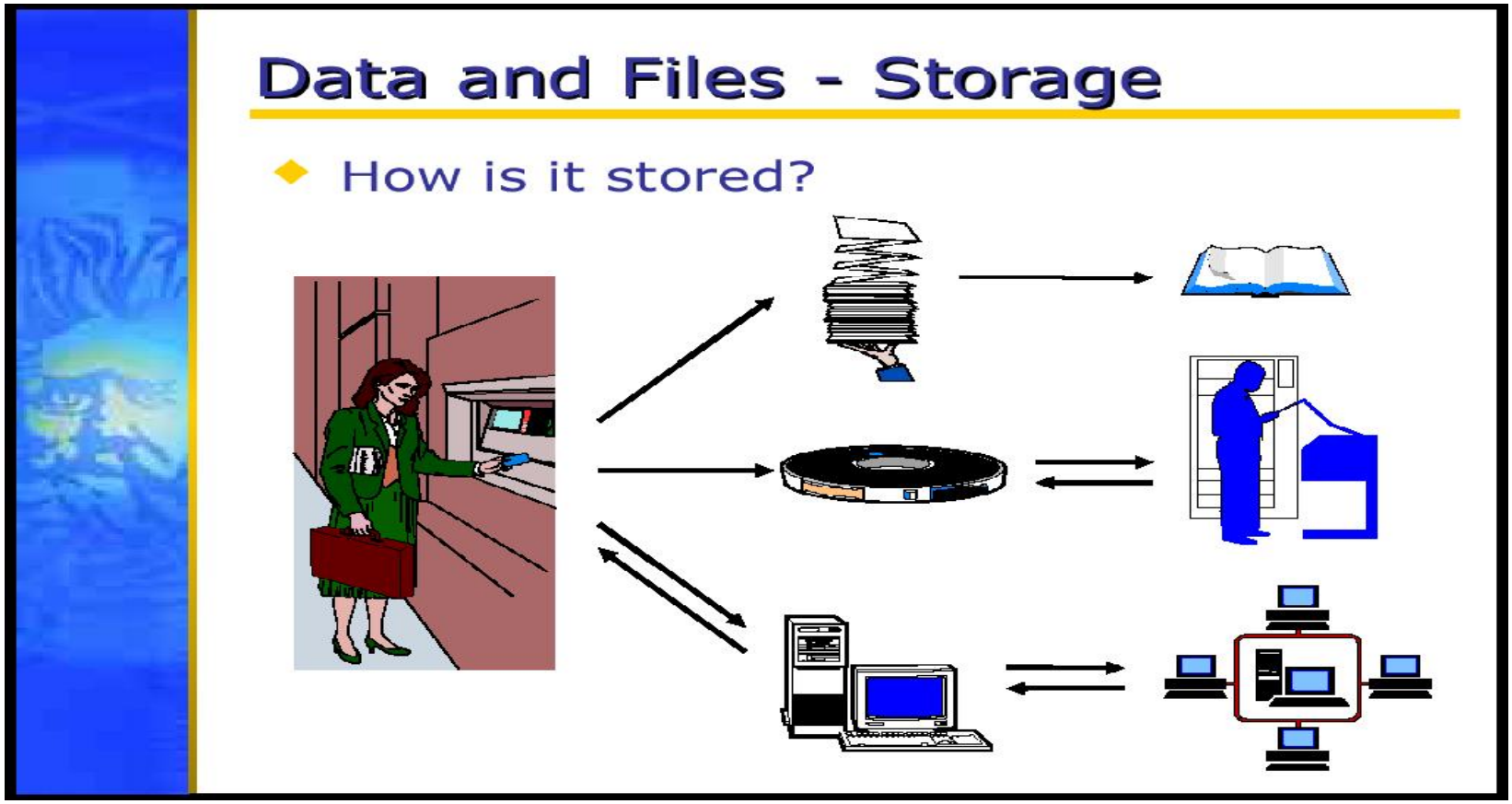
Des données ? Est ce important pour vous ?

- Assurer l'accès aux données
- Assurer la sécurité de ces données
  - Confidentialité
  - Authentification
  - Signature digitale
  - Intégrité
- Le tout efficacement, rapidement, partout etc…

Des données ? Est ce important pour vous ?

- C’est important pour vous…
- C’est impératif pour les entreprises !

Le stockage / l’accès



Un point dans le temps

- 1950-1960
  - Des fichiers séquentiels, du ‘batch’
- 1960 – 1970
  - Le début des bases de données hiérarchiques
- 1970 – 1980
  - La naissance du modèle relationnel
- Début des années 90
  - Sql, l’aide à la décision
- Fin des années 90
  - Croissance du volume des données, Internet, modèle multi tiers

Les limites à l'utilisation des fichiers (1)

- L'utilisation de fichiers impose à l'utilisateur de connaître :
  - le mode d'accès (séquentielle, indexée, …)
  - la structure physique des enregistrements
  - et la localisation des fichiers qu'il utilise afin de pouvoir accéder aux informations dont il a besoin.

- Pour des applications nouvelles, l'utilisateur devra obligatoirement écrire de nouveaux programmes et il pourra être amené à créer de nouveaux fichiers qui contiendront peut-être des informations déjà présentes dans d'autres fichiers.

- Toute modification de la structure des enregistrements (ajout d'un champ par exemple) entraîne la réécriture de tous les programmes qui manipulent ces fichiers.

Les limites à l'utilisation des fichiers (2)

- De telles applications sont
  - rigides,
  - contraignant
  - longues et coûteuse à mettre en œuvre

- Les données associées sont :
  - mal définies et mal désignées,
  - redondantes
  - peu accessibles de manière ponctuelle
  - peu fiables

Les limites à l’utilisation des fichiers (3)

- La prise de décision est une part importante de la vie d’une société. Mais elle nécessite d’être bien informé sur la situation et donc d’avoir des informations à jour et disponibles immédiatement.

- Les utilisateurs, quant à eux, ne veulent plus de systèmes d’information constitués d’un ensemble de programmes inflexibles et de données inaccessibles à tout non spécialiste; ils souhaitent des systèmes d’informations globaux, cohérents, directement accessibles (sans qu’ils aient besoin soit d’écrire des programmes soit de demander à un programmeur de les écrire pour eux) et des réponses immédiates aux questions qu’ils posent.

Les limites à l'utilisation des fichiers (4)

- Redondance des données et incohérences
- Isolation des données et accessibilité
- Un accès aux données = un programme
- Atomicité et environnement multi utilisateurs
- Sécurité et protection des données

Les limites à l'utilisation des fichiers (5)

- Source des difficultés avec les fichiers
  - Le modèle des données est intégré dans les programmes
  - Absence de contrôle pour l'accès et la manipulation des données

Notions de bases
Définition intuitive d'une BD (1)

- Définition intuitive : on peut considérer une **Base de Données** (BD) comme une grande quantité de données (ou ensemble d'informations), centralisées ou non, servant pour les besoins d'une ou plusieurs applications, interrogeables et modifiables par un groupe d'utilisateurs travaillant en parallèle.

- Exemples d'application
  - Système Socrate : SNCF
  - Annuaire électronique
  - Catalogue électronique d'une bibliothèque

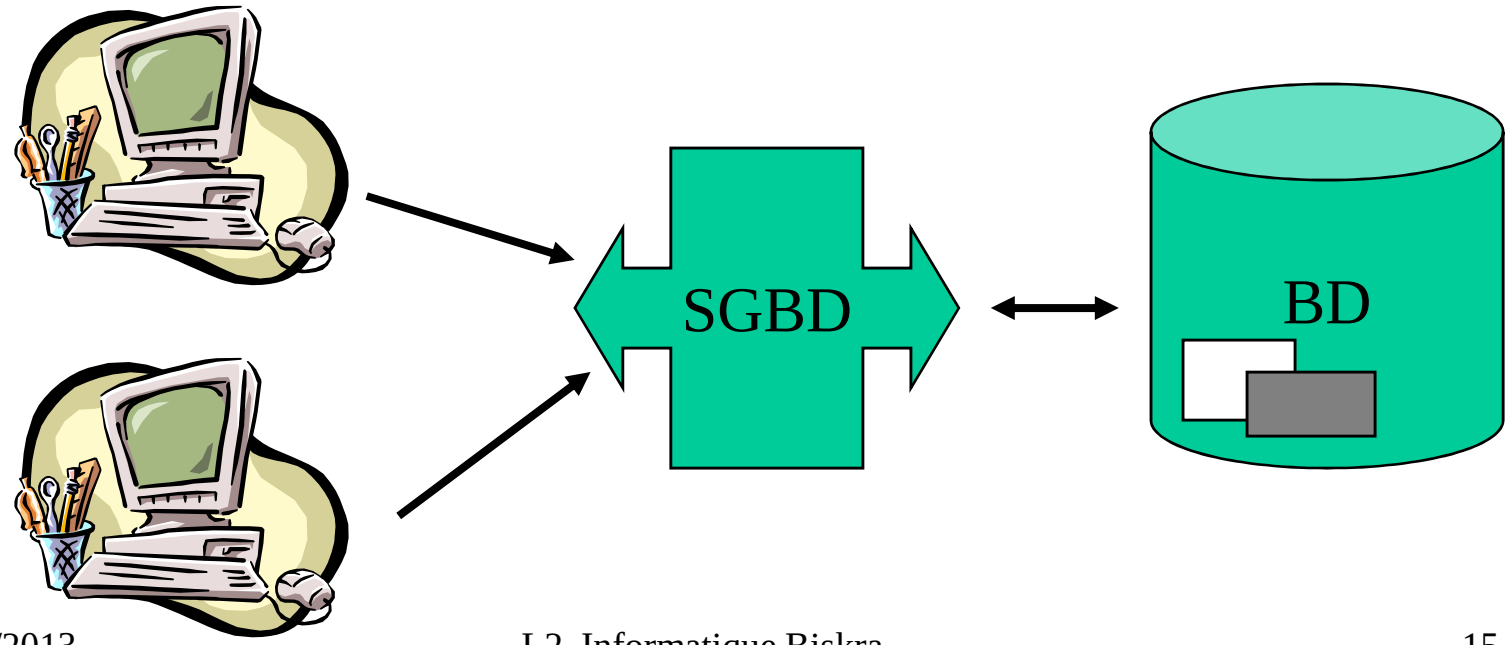
Notions de bases
Définition intuitive d'une BD (2)

- **Définition** ; une base de données est un ensemble structuré de données (1) enregistrées sur des supports accessibles par l'ordinateur (2) pour satisfaire simultanément plusieurs utilisateurs (3) de manière sélective (4) en un temps opportun (5).

  - (1) : Organisation et description de données
  - (2) : Stockage sur disque
  - (3) : Partage des données
  - (4) : Confidentialité
  - (5) : Performance

SGBD (1)

Définition : Le logiciel qui permet d'interagir avec une BD est Système de Gestion de Base de Données (SGBD)

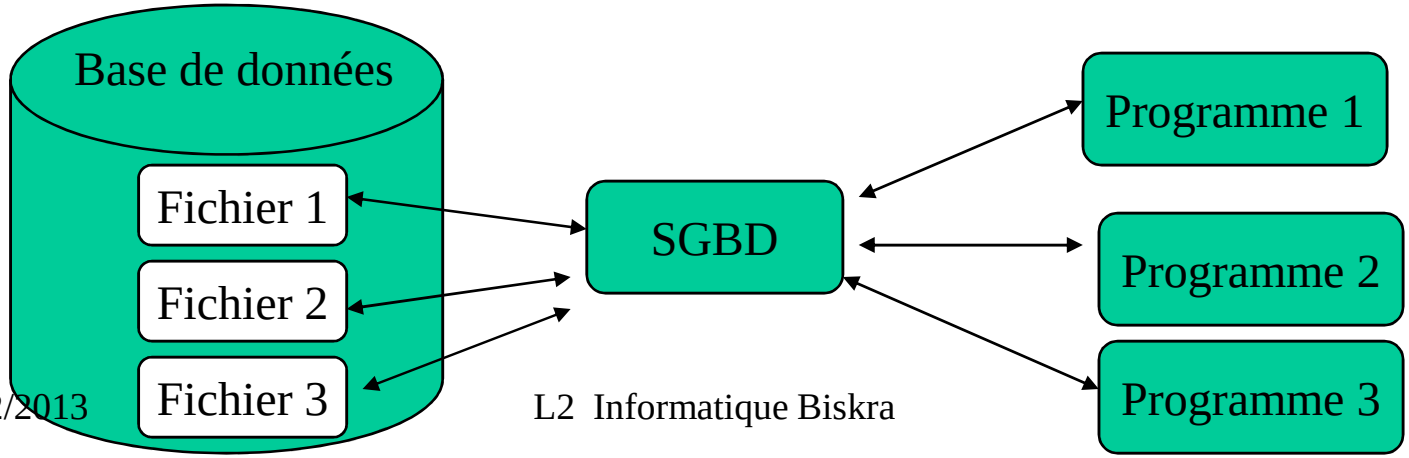


SGBD (2)

Un SGBD est un intermédiaire entre les utilisateurs et les fichiers physiques

Un SGBD facilite

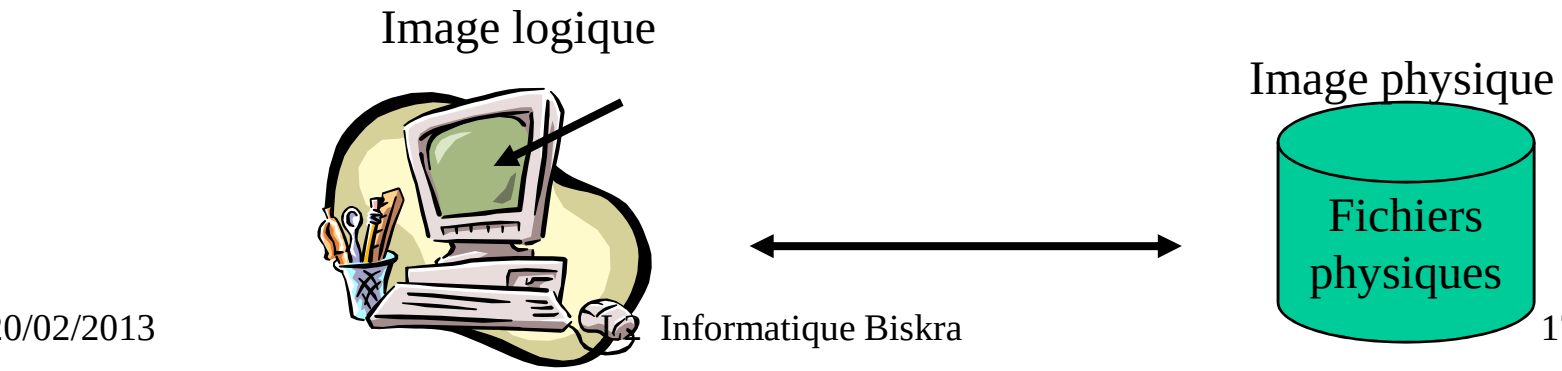
- la gestion de données, avec une représentation simple sous forme de table par exemple
- la manipulation de données. On peut insérer, modifier les données et les structures sans modifier les programmes qui manipulent la base de données



Objectifs des SGBD (1)

Faciliter la représentation et la description de données

- Indépendance physique (1) : Plus besoin de travailler directement sur les fichiers physiques (tels qu'ils sont enregistrés sur disque). Un SGBD nous permet de décrire les données et les liens entre elles d'une façon logique sans se soucier du comment cela va se faire physiquement dans les fichiers. On parle alors d'image logique de la base de données, (ou aussi description logique ou conceptuelle ou encore de schéma logique). Ce schéma est décrit dans un modèle de données par exemple le modèles de tables, appelé le modèle relationnel.



Objectifs des SGBD (2)

- Indépendance physique (2) : La manipulation des données doit être faciliter en travaillant directement sur le schéma logique. On peut insérer, supprimer, modifier des données directement sur l'image logique. Le SGBD va s'occuper de faire le travail sur les fichiers physiques.
- Indépendance logique : Un même ensemble de données peut être vu différemment par des utilisateurs différents. Toutes ces visions personnelles des données doivent être intégrés dans une vision globale.
- Manipulations des données par des non informaticiens. Il faut pouvoir accéder aux données sans savoir programmer ce qui signifie des langages « quasi naturels ».
- Efficacité des accès aux données : Ces langages doivent permettre d'obtenir des réponses aux interrogations en un temps « raisonnable ». Il doivent donc être optimisés et, entre autres, il faut un mécanisme permettant de minimiser le nombre d'accès disques. Tout ceci, bien sur, de façon complètement transparente pour l'utilisateur.

Objectifs des SGBD (3)

- Administration centralisée des données : Des visions différentes des données (entre autres) se résolvent plus facilement si les données sont administrées de façon centralisée.
- Cohérence des données. Les données sont soumises à un certain nombre de contrainte d'intégrité qui définissent un état cohérent de la base. Elles doivent pouvoir être exprimées simplement et vérifiées automatiquement à chaque insertion, modification ou suppression de données, par exemple :
  - l'âge d'une personne supérieur à zéro
  - Salaire supérieur à zéro
  - Etc

  Dés que l'on essaie de saisir une valeur qui ne respecte pas cette contrainte, le SGBD le refuse.

Objectifs des SGBD (4)

- Non redondance des données : Afin d'éviter les problèmes lors des mises à jour, chaque donnée ne doit être présente qu'une seule fois dans la base.
- Partageabilité des données : Il s'agit de permettre à plusieurs utilisateurs d'accéder aux mêmes données au même moment. Si ce problème est simple à résoudre quand il s'agit uniquement d'interrogations et quand on est dans un contexte mono-utilisateur, cela n'est plus le cas quand il s'agit de modifications dans un contexte multi-utilisateurs. Il s'agit alors de pouvoir :
  - Permettre à deux (ou plus) utilisateurs de modifier la même donnée « en même temps »;
  - Assurer un résultat d'interrogation cohérent pour un utilisateur consultant une table pendant qu'un autre la modifie.

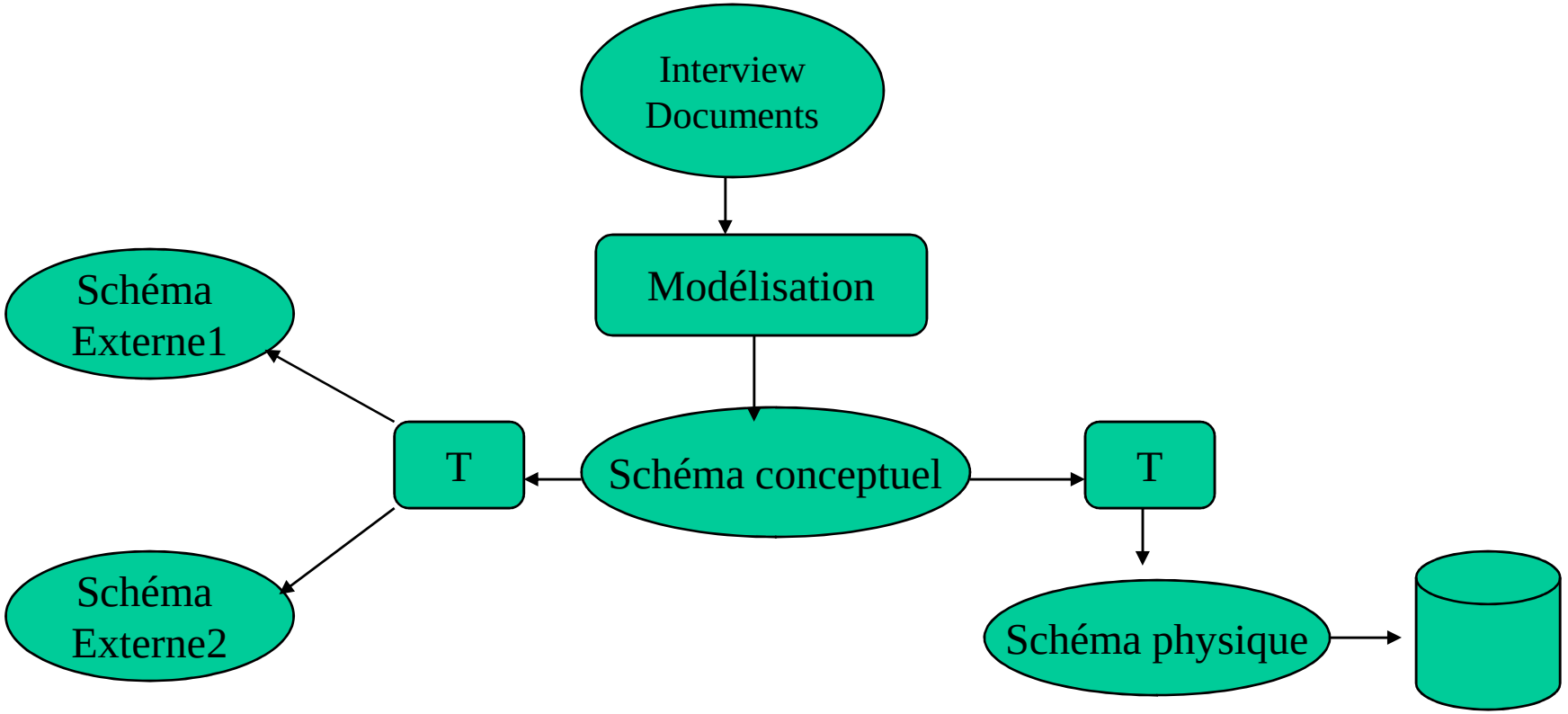
Objectifs des SGBD (5)

- Sécurité des données. Les données doivent pouvoir être protégées contre les accès non autorisés. Pour cela, il faut pouvoir associer à chaque utilisateur des droits d'accès aux données.
- Résistance aux pannes : Que se passe-t-il si une panne survient au milieu d'une modification, si certains fichiers contenant les données deviennent illisibles? Les pannes, bien qu'étant assez rares, se produisent quand même de temps en temps. Il faut pouvoir, lorsque l'une d'elles arrive, récupérer une base dans un état « sain ». Ainsi, après une panne intervenant au milieu d'une modification deux solutions sont possibles : soit récupérer les données dans l'état dans lequel elles étaient avant la modification, soit terminer l'opération interrompue.

Trois Fonctions d'un SGBD

- Description des données : codification structuration, grâce à un Langage de Description de Données (LDD)
- Manipulation et restitution des données (insertion, mise à jour, interrogation)
  - Mise en œuvre à l'aide d'un Langage de Manipulation de Données (LMD)
  - S.Q.L. (Structures Query Langage) : Langage standard
- Contrôle (partage, intégrité, confidentialité, sécurité)

Définition et description des données
3 niveaux de description



Définition et description des données niveau logique (conceptuel)

- Permet la description
  - Des objets : exemple OUVRAGES, ETUDIANTS
  - Des propriétés des objets (attributs) : exemple Titre de OUVRAGES
  - Des liens entre les objets : un OUVRAGE peut être emprunté par un ETUDIANT
  - Des contraintes : le nombre d'exemplaires d'un OUVRAGE est supérieur à zéro
- Cette description est faite selon un modèle de données.
- Un modèle de données est un ensemble de concepts permettant de décrire la structure d'une base de données. La plupart des modèles de données incluent des opérations permettant de mettre à jour et questionner la base. Le modèle de données le plus utilisé est le modèle relationnel
- Cette description va donner lieu à un schéma de base de données. Un schéma de base de données se compose d'une description des données et de leurs relations ainsi que d'un ensemble de contraintes d'intégrité.

Définition et description des données niveau physique

- Description informatique des données et de leur organisation : en terme de fichiers, d'index, de méthodes d'accès, …

- Passage du modèle logique au modèle physique tend à être assisté par le SGBD : transparent et/ou semi-automatique

- Objectifs : optimiser les performances

Définition et description des données niveau externe

- Description des données vues par un utilisateur (ou un groupe d'utilisateurs)
  - Objectifs : simplification, confidentialité
  - Exemple : OUVRAGES édité par des éditeurs français

Manipulation et restitution des données

- Afin de réaliser les opérations suivantes
  - Insertion : saisir des données
  - Supprimer
  - Modifier
  - Interroger : rechercher des données via des requêtes

  La manipulation des données est mise en œuvre à l'aide d'un Langage de manipulation de Données (LMD). SQL (Structured Query Language) est le langage standard de manipulation de BD

Contrôles réalisés par le SGBD

- Partage de données : accès à la même information par plusieurs utilisateurs en même temps. Le SGBD inclut un mécanisme de contrôle de la concurrence basé sur des techniques de verrouillage des données (pour éviter par exemple qu'on puisse lire une information qu'on est en train de mettre à jour)
- Intégrité des données grâce à la définition de contraintes sur les données. Le SGBD veille à ce que toutes les contraintes soient vérifiées à chaque insertion, suppression, ou modification d'une donnée.
- Confidentialité : plusieurs utilisateurs peuvent utiliser en même temps une base de données, se pose le problème de la confidentialité des données. Des droits doivent être gérés sur les données, droits de lecture, mise à jour, création; … qui permettent d'affiner.
- Sécurité : une base de données est souvent vitale dans le fonctionnement d'une organisation, et il n'est pas tolérable qu'une panne puisse remettre en cause son fonctionnement de manière durable. Les SGBD fournissent des mécanismes pour assurer cette sécurité.

Utilisateurs des SGBD

Les différents rôles que doivent jouer un individu ou un groupe d'individus pour concevoir, créer, mettre en œuvre et exploiter une base de données.

- ***L'administrateur de la base de données :*** *Il est chargé* de décrire les entités de la base de données et indiquer les liaisons existant entre ces entités, ceci au moyen du DDL offert par le SGBD.
- ***Le programmeur d'application :*** Il est chargé d'élaborer les programmes pour exploiter la base de données en fonction de la description qui a été faite par l'administrateur d'application. Le programmeur d'application utilise le LMD offert par le SGBD ainsi que d'autres sous-programmes conservés généralement dans une librairie (i.e. bibliothèque de sous-programmes).

Utilisateurs des SGBD

- ***L'utilisateur :*** Il s'agit de caractériser ici la personne qui se sert simplement de la base de données et qu'on appelle couramment *l'utilisateur final (End User en anglais).*
- *Exemple :*

  Dans une agence de réservation de billets d'avion, la personne qui tape sur son terminal quelques commandes pour effectuer une réservation est une utilisatrice au même titre qu'un chef d'entreprise qui lui aussi demande de temps en temps à une base de données de son entreprise un certains nombre d'informations reflétant l'état de son entreprise (produits non vendus, commandes en attente, etc.).

Modèles de SGBD

- Quelques modèles logiques :
  - Modèle hiérarchique
  - Modèle réseau
  - Modèle relationnel
  - Modèle objet
- Quelques SGBD (relationnels du marché)
  - Micro : ACCESS, Paradox, Dbase, PostSQL, MySQL, …
  - Gros système : DB2, ORACLE, SYBASE, …

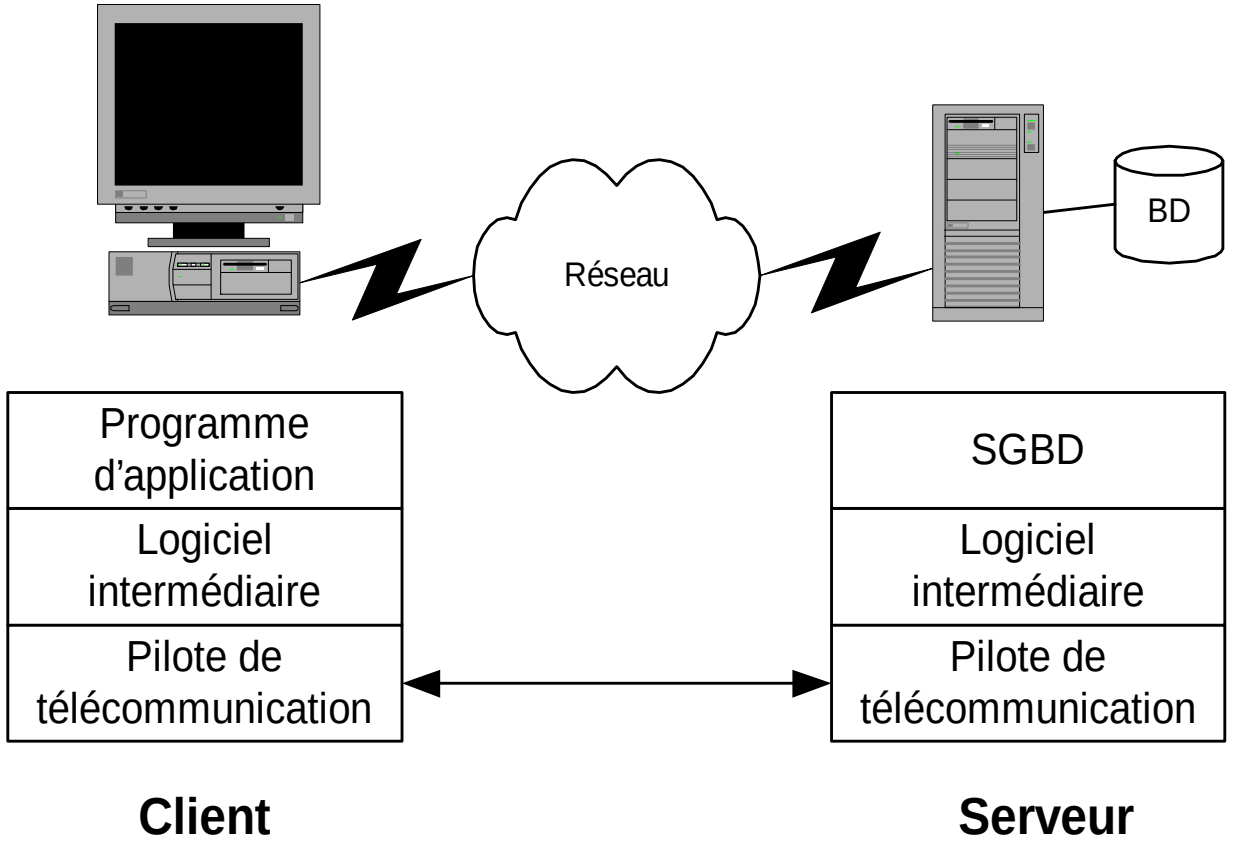
L’architecture des SGBD

- Basée sur une architecture Client-Serveur
  - Données sur le serveur partagées entre N clients
  - Interfaces graphiques sur la station de travail personnelle
  - Communication par des protocoles standardisés
  - Clients et serveurs communiquant par des requêtes avec réponses

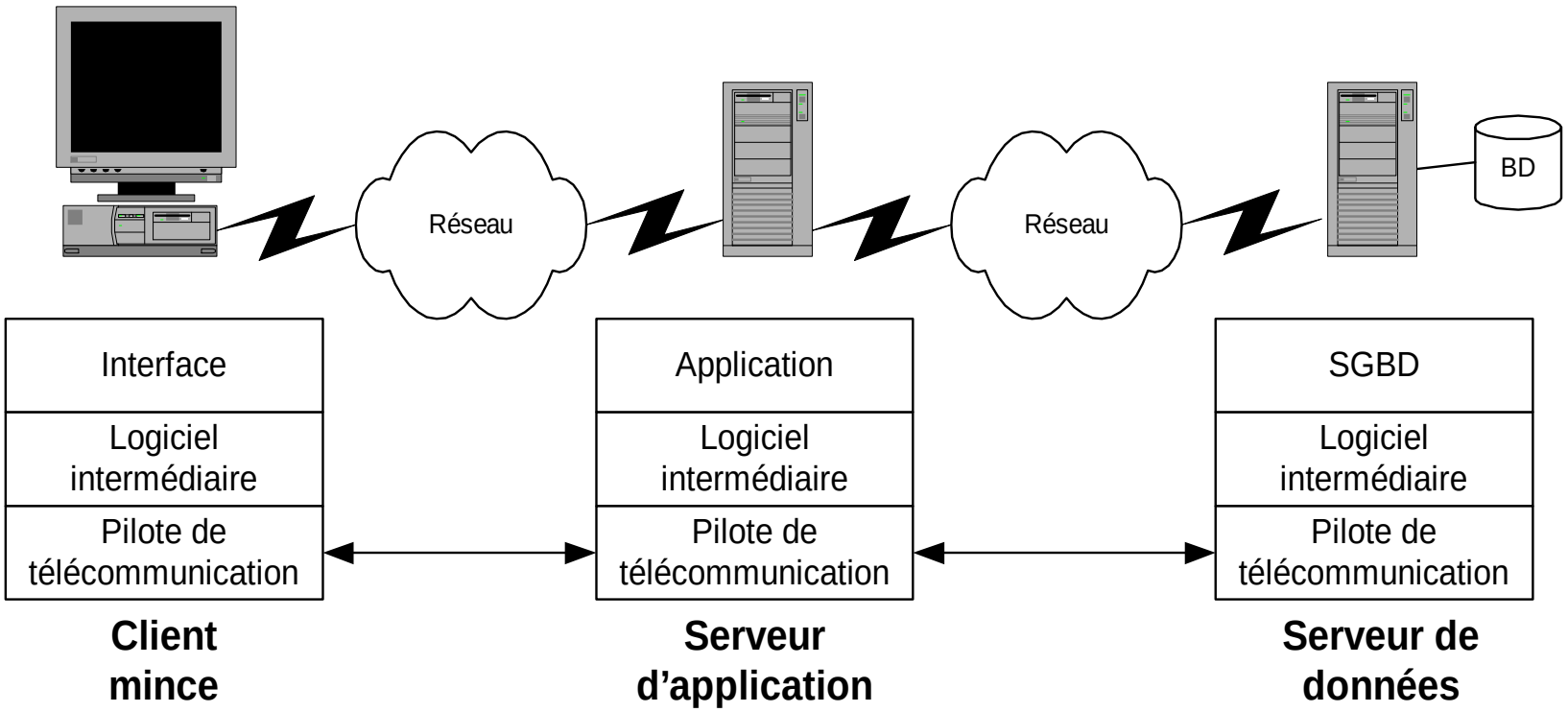
Architecture

- ***Architecture centralisée***
  - programme d'application et SGBD sur même machine (même site)
  - premiers systèmes
- ***Architecture du type client-serveur* (*client-server architecture*)**
  - programme d'application = *client*
    - interface (« GUI ») + traitement du domaine d 'application
  - SGBD = *serveur* (*de données « data server »*)
  - machines (sites) différentes
  - *deux couches*, *niveaux*, *strates* (“*two tier* ”)

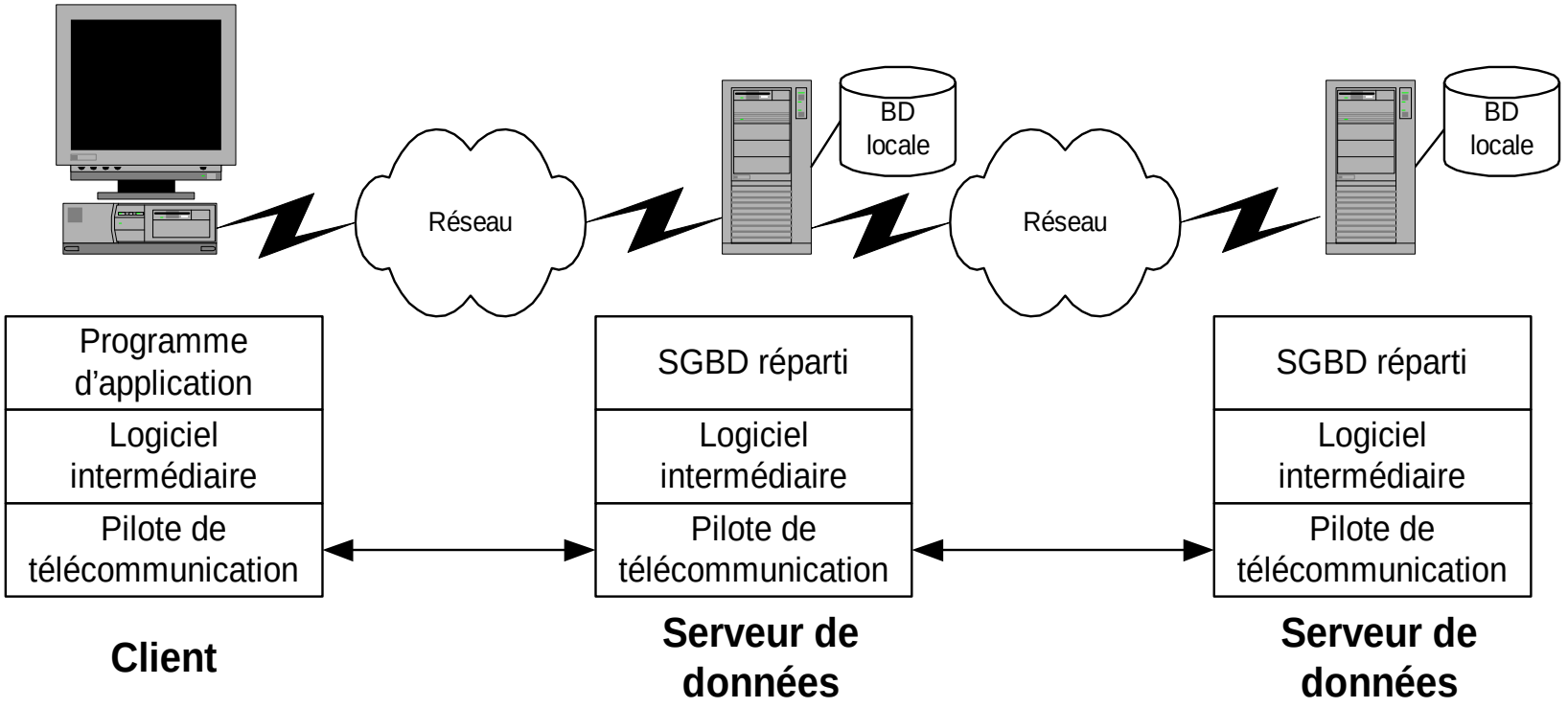
Architecture client / serveur



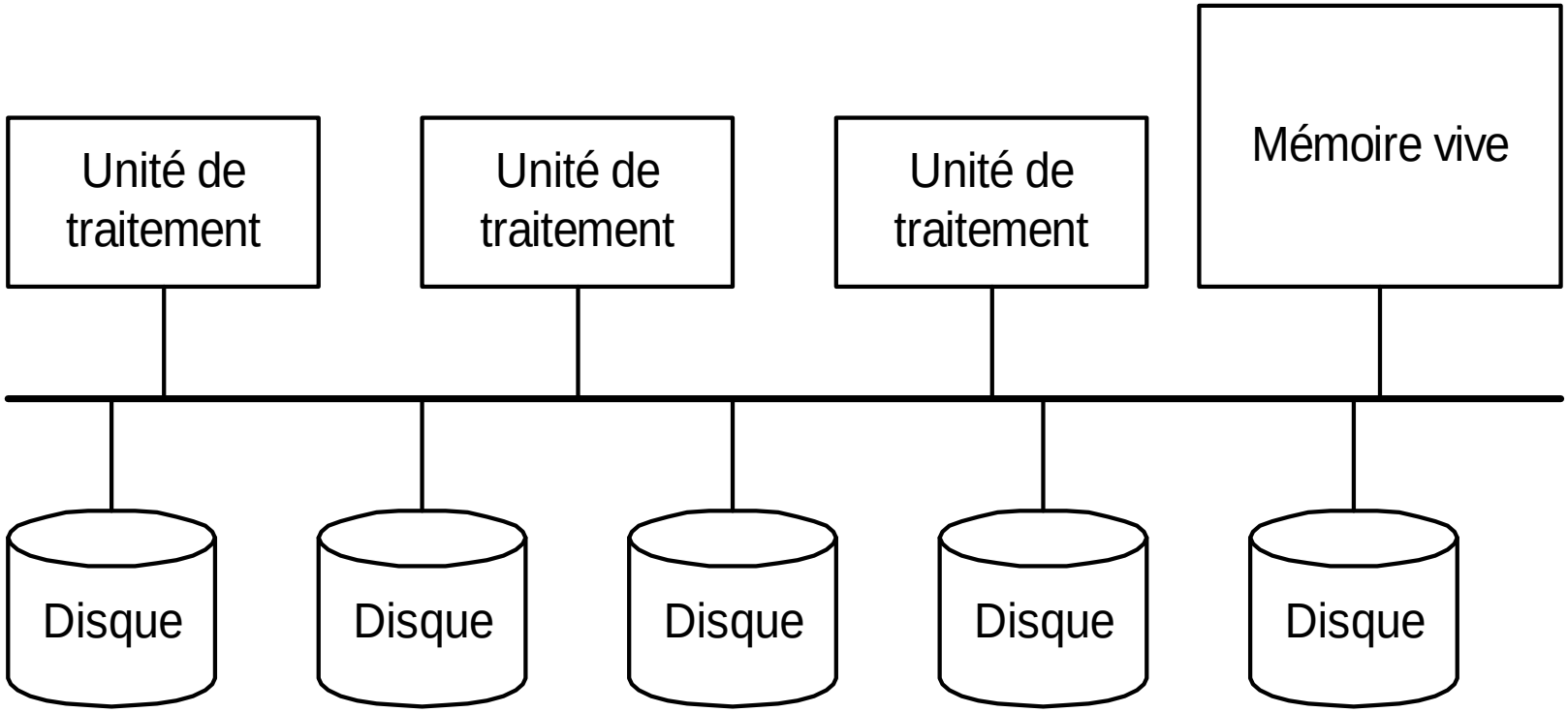
Architecture 3 tiers



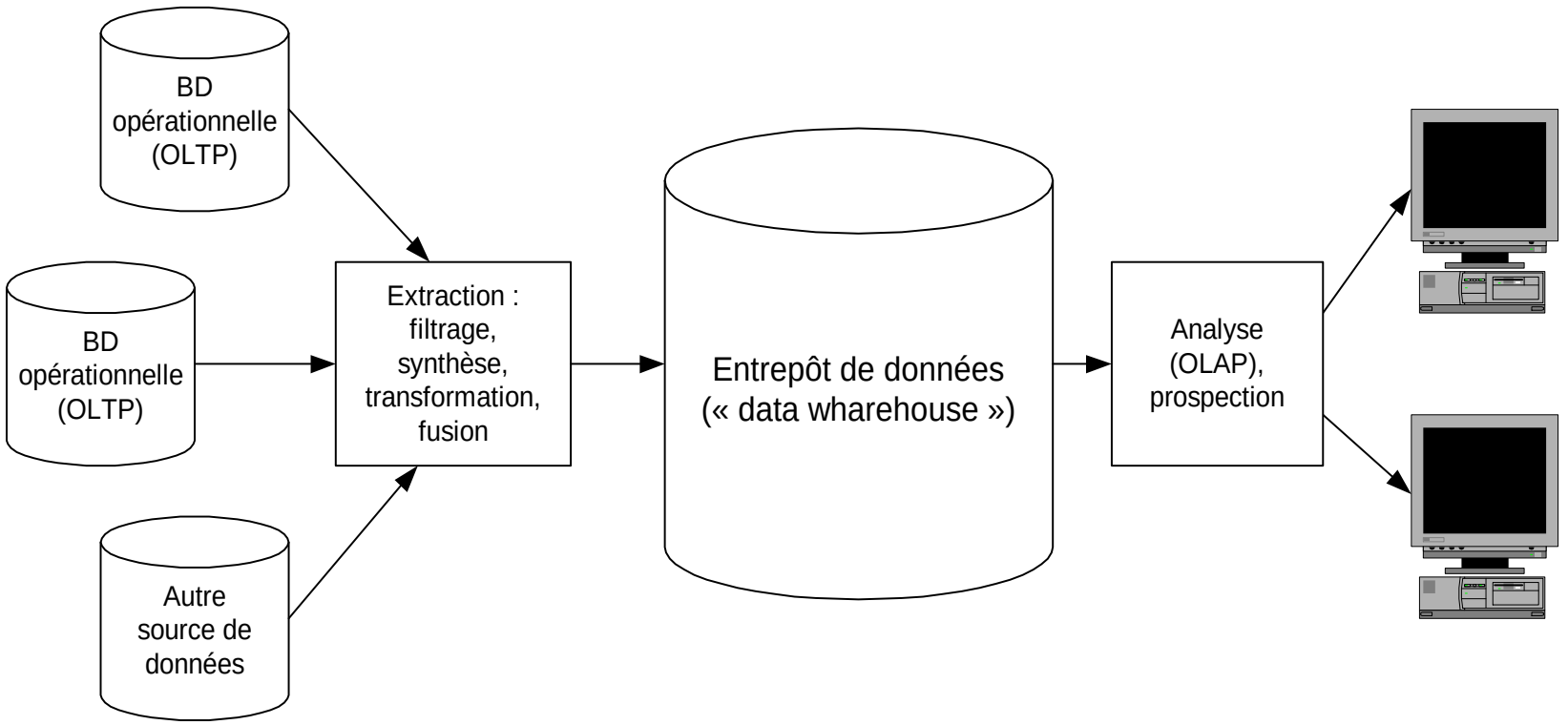
Base de données distribuées



Base de données parallèles



Entrepôt de données



Entrepôt de données

- ***Base de données opérationnelle***
  - traitement des données quotidiennes et récentes
  - OLTP (“ *On Line Transaction Processing* ”).
- ***Entrepôt de données* (*data wharehouse*)**
  - grand volume de données historiques extraites de bases opérationnelles pour le support à la prise de décision
  - OLAP (“ *On Line Analytical Processing* ”)
- ***Prospection de données* , ou *forage, fouille, exploration de données, ou découverte de connaissances dans les BD* (*data mining, analysis, dredging, archeology, knowledge discovery in databases - KDD)***
  - extraction non triviale d’informations implicites, inconnues et utiles
  - apprentissage machine , statistiques

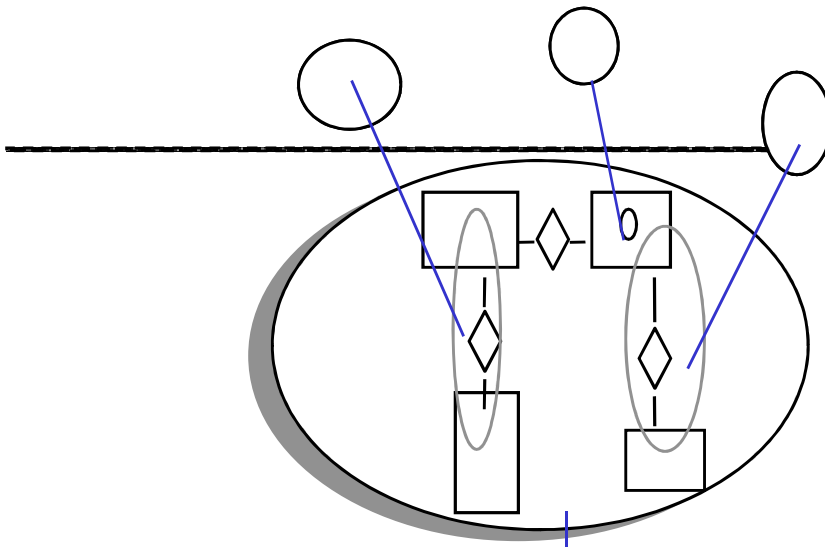
Chapitre 02

Modélisation de bases de données : Le modèle relationnel

Introduction

- Rappel chapitre 1
- C’est quoi un modèle ?
- Type de modèle :
  - Modèle hiérarchique
  - Modèle réseau
  - Modèle objet
  - Modèle relationnel

Les niveaux des BD



Vues externes
chaque vue représente
la partie de la bd relative
à chaque utilisateur / application

Schéma conceptuel
vue globale de la base :
les données stockées
et les relations parmi les données

R_1(A number(10), B string, …, primary key (A), ...)
….
R_n (X number(15), Y string, …, primary key (X), …)

Schéma logique
Représente un modèle de données
conforme au modèle ciblé de SGBD
(par ex. SGBDR)

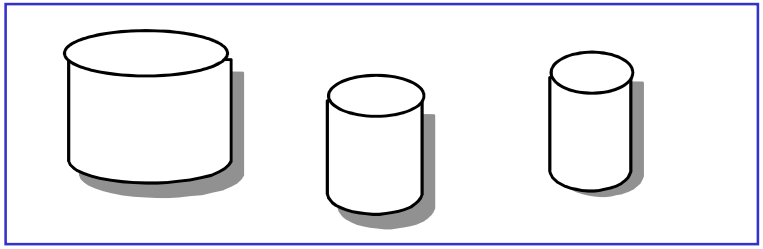


Schéma physique
décrit comment les données
sont stockées dans la bd +
les index

Caractéristiques de l'Architecture

Schéma externe

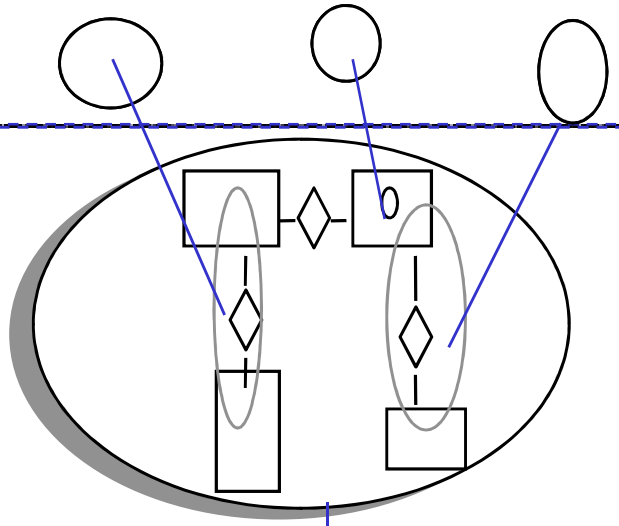


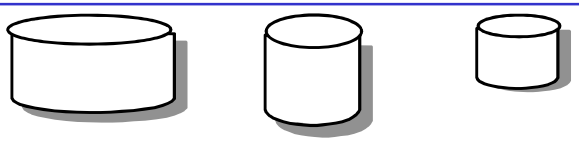
Schéma conceptuel

Indépendance logique :
Un changement du schéma conceptuel (nouvel attribut, nouvelle relation, …), n'entraîne pas de changement dans les vues existantes

Schéma logique

R_1(A number(10), B string, …, primary key (A), ...)
….
R_n (X number(15), Y string, …, primary key (X), …)

Schéma physique



Indépendance physique :
Les changements du schéma interne ne sont pas visibles aux niveaux des schémas conceptuels et logiques; conséquences sur les performances

Cfr. ANSI / SPARC 1975

Quels types d'études dans les BD et les SGBD ?

- Conception de BD

  À partir de l'analyse du contexte, recueillir les besoins :

  - Quelles informations stocker ? Comment les organiser ?
  - Quelles contraintes prendre en compte ? Comment les représenter ?
  - Quels types d'usages de ces données ?

- Programmation

  Développement et optimisation de requêtes et programmes

  Interfaces avec d'autres applications

- Implémentation de SGBD

  support de nouveaux types de données, optimiseurs, intégration de données et d'applications, interfaces utilisateurs, langages

Type de modèle

Un schéma conceptuel est donc le résultat d'un processus de modélisation fait en respectant les possibilités d'un modèle de données.

Le modèle de données est une caractéristique de tout SGBD. Il existe quatre grandes classes de modèles de données qui se distinguent par la nature des associations qu'ils permettent de modéliser.

- Les modèles hiérarchiques
- Les modèles réseaux
- Les modèles objets
- Les modèles relationnels

Efforts de standardisation et de développement des SGBD.

- CODASYL (COnference on DAta SYstems Language)
- ANSI (American National Standard Institute)
- GUIDE/SHARE (groupe d'usagers IBM)

Le groupe CODASYL

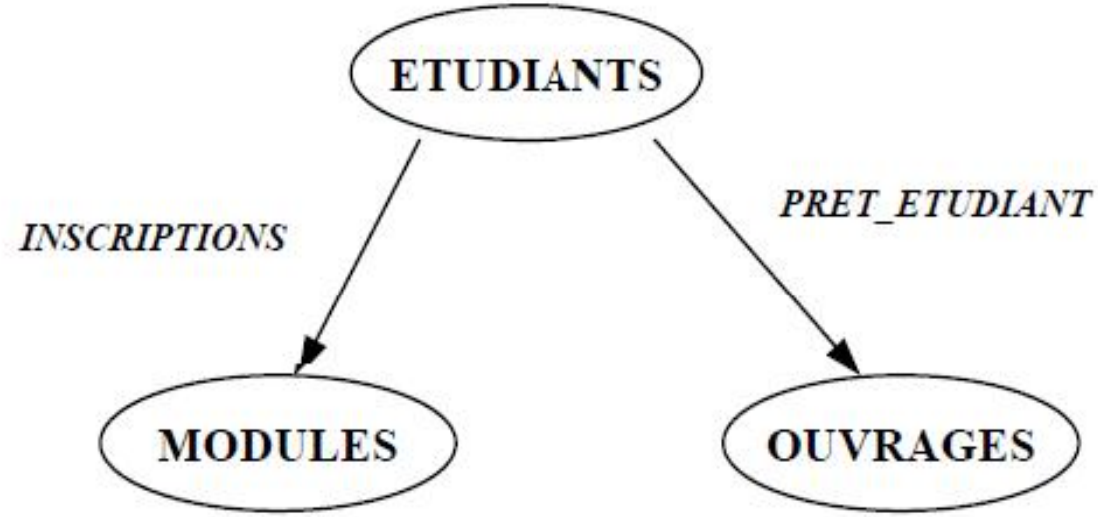
- Enregistrement : collection d'agrégats et d'atomes rangés consécutivement constituant l'unité d'échange entre la base de données et les applications.
- • SETS : Association entre un enregistrement père (propriétaire ou Owner) et un enregistrement fils (Membre ou Member).
- Plus de détails la référence 1 page 14.

Modèle hiérarchique (1)

- A l'aide du modèle hiérarchique, le schéma conceptuel peut être vu comme un graphe arborescent dont les nœuds correspondent aux classes d'objets (entités) et les arcs entre deux nœuds aux liaisons ou associations entre les entités.
- Un tel graphe possède donc un nœud racine (sur lequel n'arrive aucun arc !) et les autres nœuds sont des fils, petit-fils, etc., de cette racine.
- Avec le modèle hiérarchique, le nombre de flèches pouvant arriver sur un nœud est donc égal à un (sauf pour le nœud racine).

Modèle hiérarchique (2)

exemple de modèle hiérarchique

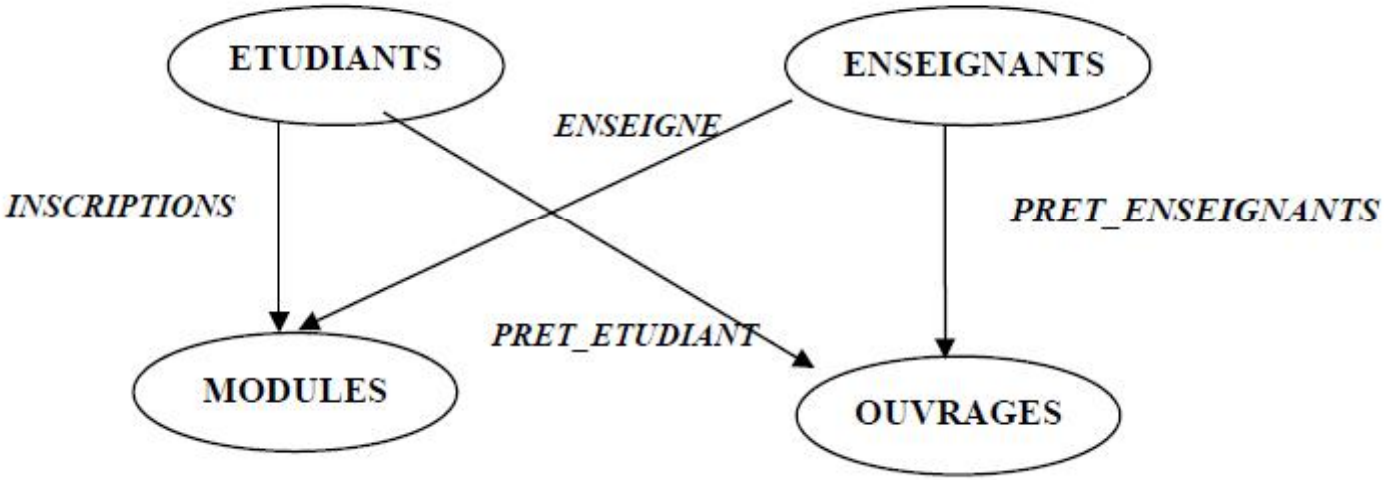


Modèle réseaux (1)

- A l'aide de ce modèle, le schéma conceptuel peut être vu comme un graphe général où les nœuds correspondent aux classes d'objets et les arcs entre deux nœuds aux associations.
- A la différence du modèle hiérarchique on peut avoir ici plusieurs arcs qui arrivent sur le même nœud. De même que la notion de nœud racine n'existe pas avec le modèle réseau.

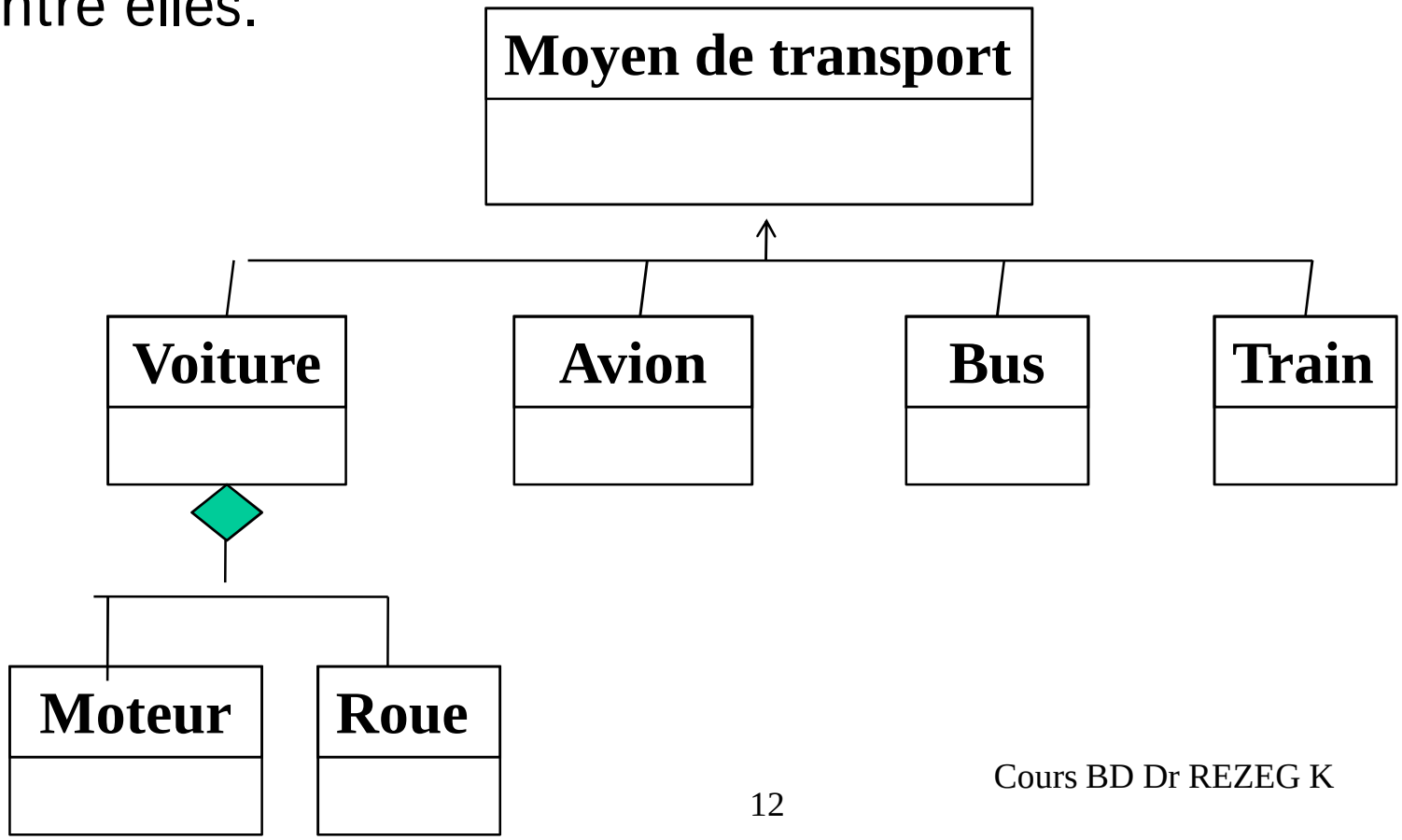
Modèle réseaux (2)

Exemple de modèle Réseau



Modèle Objet (1)

- Permet de voir une base de données comme un ensemble de classes d'objets, ayant des liens d'héritage, d'agrégation, de composition ou de simple association entre elles.



Modèle relationnel (1)

Le modèle relationnel a été introduit au début des années 70, c'est le premier modèle de bases de données indépendant des critères de stockage, il consiste à percevoir la base de données comme un ensemble de relations qui peuvent être représentées sous forme de table à deux dimensions : Les colonnes correspondent aux attributs d'une relation et les lignes correspondent aux tuples. Contrairement aux autres modèles de conception qui distinguent entre les concepts d'entité et de lien ou relation, ce modèle utilise un seul concept qui est la relation, ce qui lui rend plus facile à utiliser même pour les utilisateurs ayant plus ou moins de connaissance en informatique.

Ce modèle est basé essentiellement sur une théorie mathématique « la théorie des ensembles »

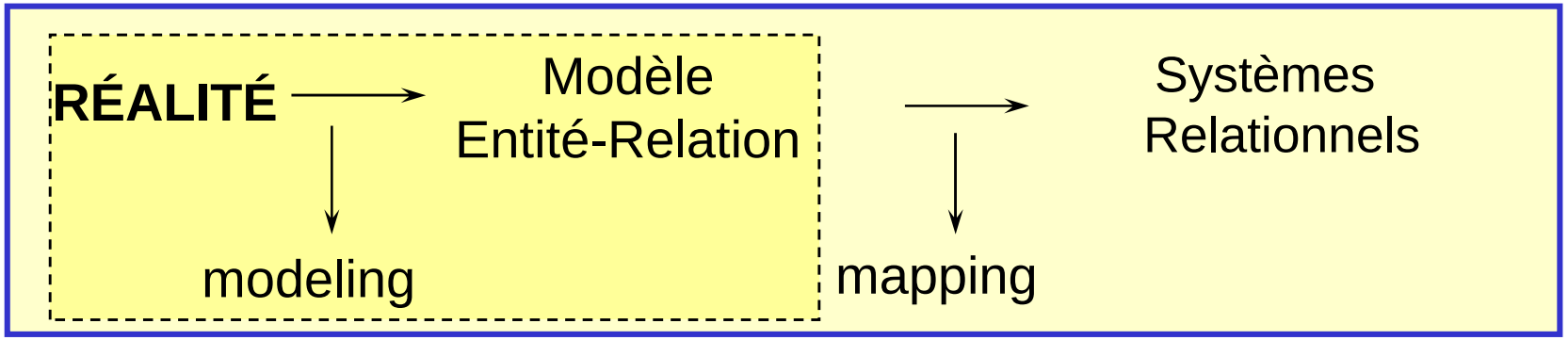
2 - Modèle Entités - Relations

- Contenu
  - Concepts de base
    - Entités, attributs
    - Relations
    - Clés
    - Multiplicités, cardinalités
  - Concepts avancés
    - Héritage
    - Agrégation, Composition
    - Entités faibles

Objectif du modèle de données

- Apporter une interprétation du contexte applicatif
  - en soulignant les aspects fondamentaux
  - en négligeant les détails
- Identifier les données gérées par la base de données et leur relations
- Produire un diagramme, suivant un formalisme,

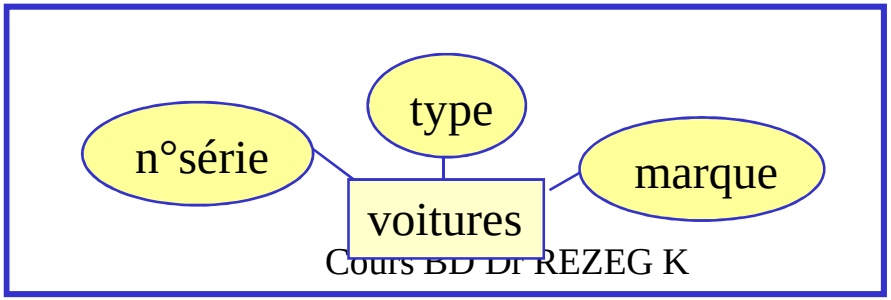
  en vue de convertir le modèle de données en schéma de base relationnelle



Entité -Attributs

- Entité
  - "une chose" qui existe et qui peut être distinguée de façon unique.
    Ex.: un étudiant, une voiture, une banque
  - abstraite ou concrète
- Attribut
  - propriété d'une entité
  - prend des valeurs simples, par exemple entiers ou chaînes de caractères (domaine d'attribut)

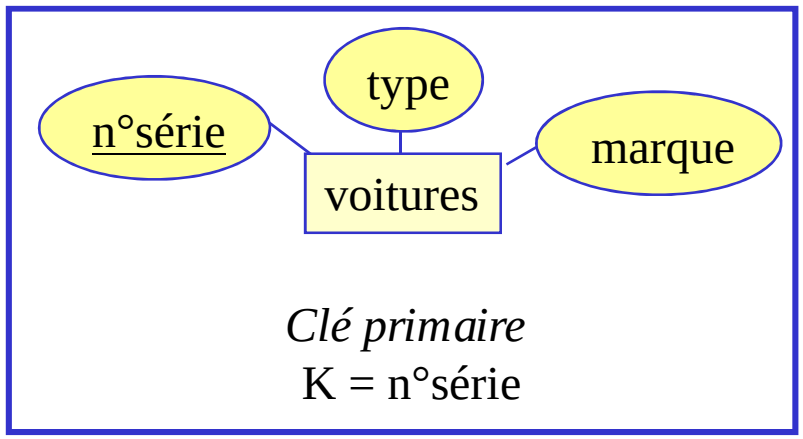
Ensemble d'Entités	**Attributs**	**Domaines**
Voitures	n°série	entier (12)
	marque	chaîne de car. (8)
	type	chaîne de car. (10)

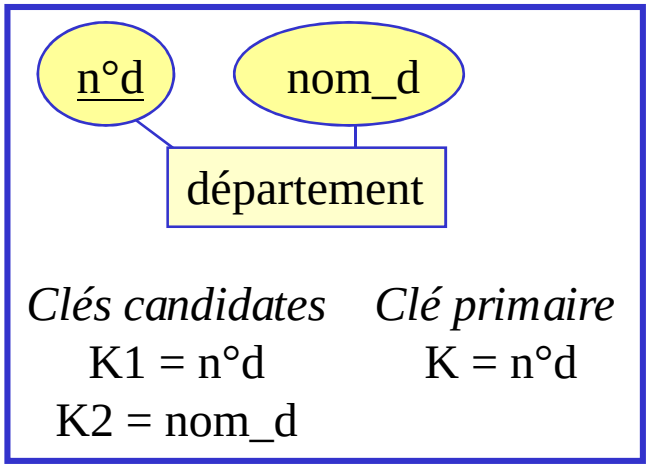


Représentation sous forme de diagramme

Clé des ensembles d'entités

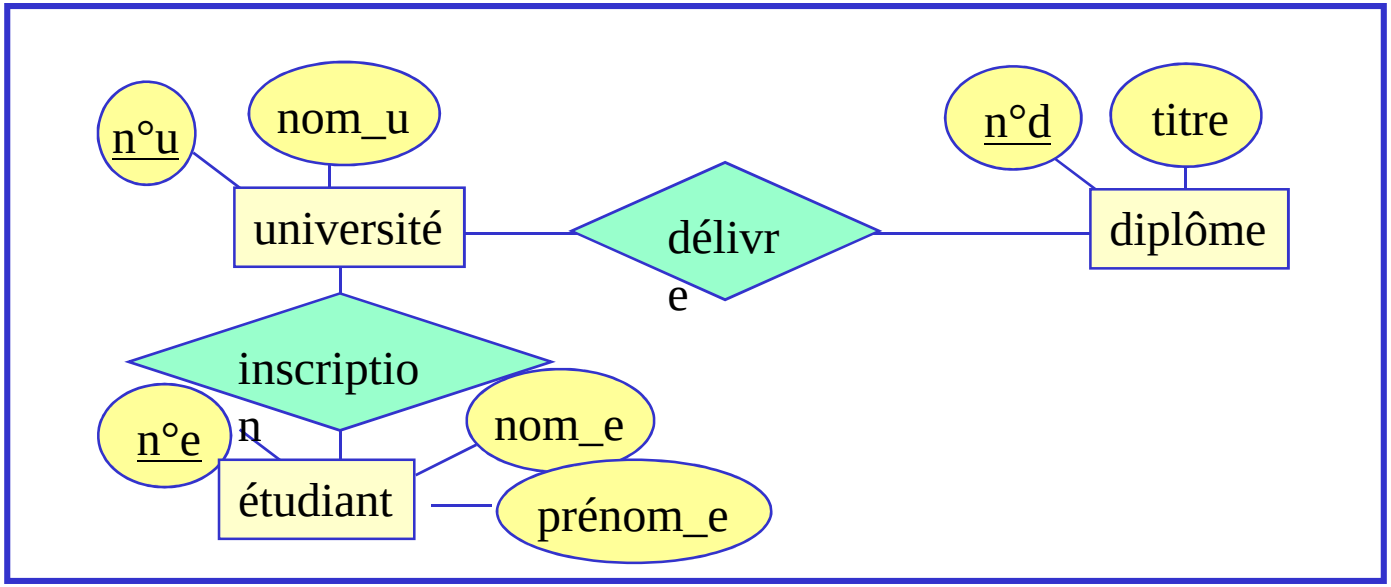
- clé candidate: un ensemble minimal d'attributs qui identifie de façon unique une occurrence d'entité
- clé primaire: une clé candidate choisie pour identifier de façon unique chaque occurrence d'entité
- clé composée: une clé candidate composée de deux ou plusieurs attributs

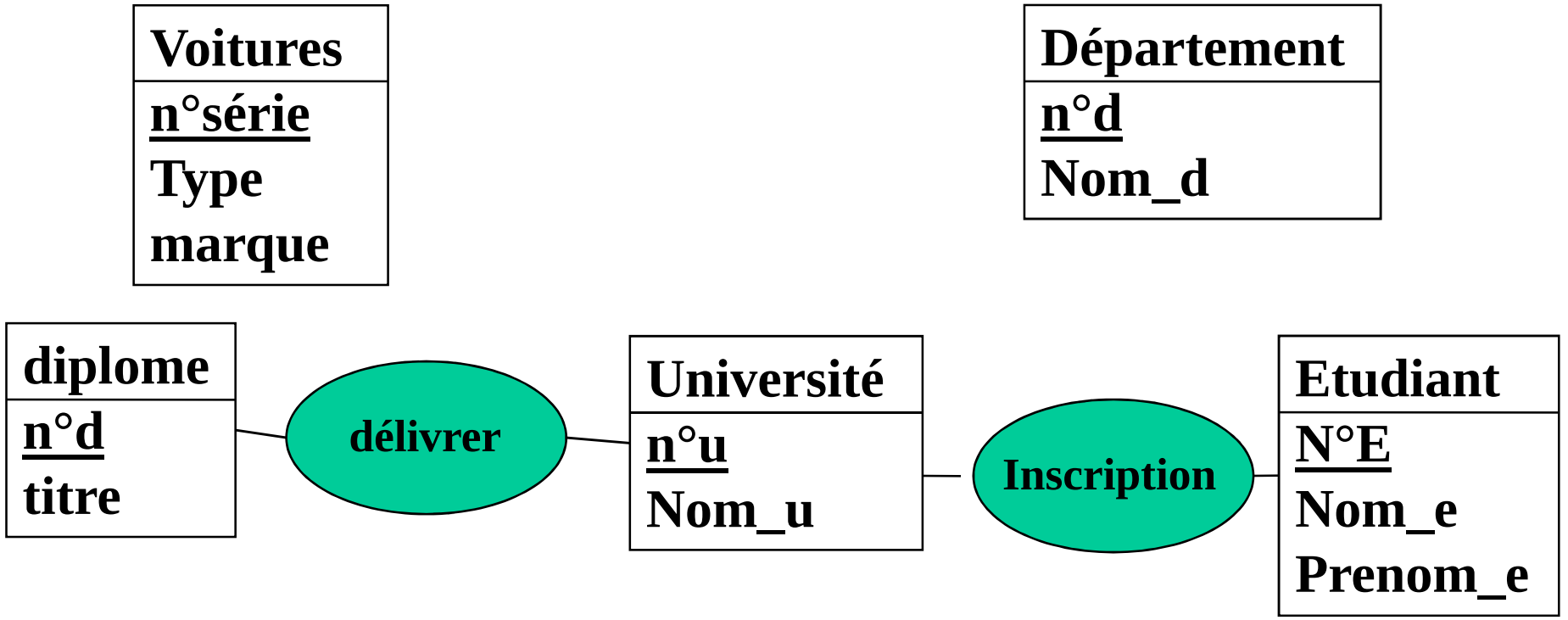




Relations

- Une relation relie deux ou plusieurs ensembles d'entités
- Ex.:
  - Des universités délivrent des diplômes
  - Des étudiants sont inscrits dans des universités




Voitures
n°série
Type
marque
Département
n°d
Nom_d
diplome
n°d
titre
délivrer
Université
n°u
Nom_u
Inscription
Etudiant
N°E
Nom_e
Prenom_e

Ensemble de relations

- La "valeur" d'une relation est l'ensemble des listes des entités réellement associées par la relation. Chaque liste est obtenue en correspondance des ensembles d'entités en relation.

  Exemple: valeur de la relation "délivre"

n°u	n°d
u_1	d_2
u_1	d_3
u_1	d_5
….	…
u_{100}	d_1
u_{100}	d_2

- l'université identifiée par u_1 délivre les diplômes identifiés par d_2, d_3 et d_5,
…
l'université identifiée par u_{100} délivre les diplômes identifiés par d_1 et d_2

Ensemble de relations (2)

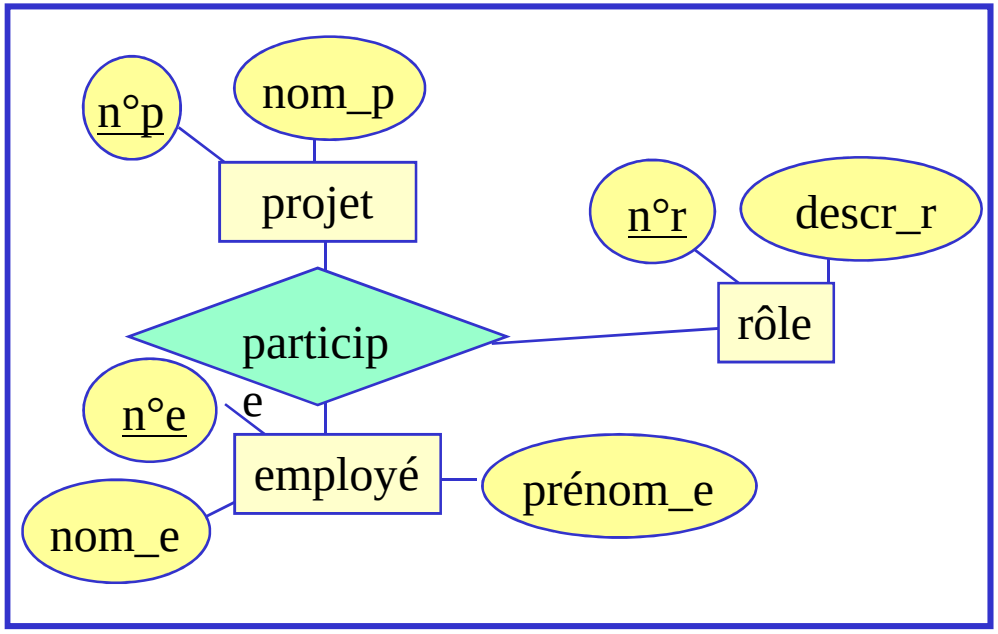
- Soit r une relation entre k ensembles d'entités E_1, E_2, ..., E_k. Un ensemble d'entités peut apparaître plus d'une fois dans la liste.

- Soit un k-uplet $(e_1, \ldots e_k) \in r$
  - $e_1 \in E_1$, ... $e_k \in E_k$ sont associés par la relation r
  - e_1, ... e_k est dit tuple de r
  - k est le degré de la relation
    - K = 1 : relation unaire (ou récursive, sur un même ensemble d'entités)
    - K = 2 : relation binaire
    - K = 3 : relation ternaire
    - ...
    - K = n : relation n-aire

Relations n-aires

- Une relation connectant plus de 2 ensembles d'entités permet de restreindre les combinatoires entre les valeurs des entités en relation

  Ex.: des employés participent à des projets avec des rôles

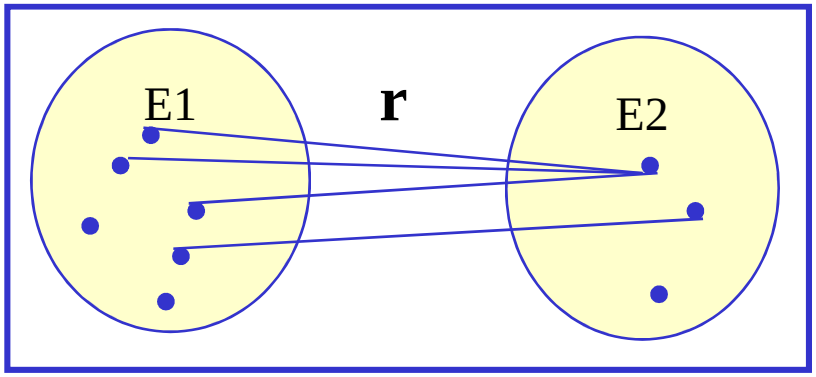


n°p	n°e	n°r
p1	e2	r1
p1	e3	r2
p1	e5	r3
…	…	…
p2	e2	r2
p2	e3	r1

Type des relations binaires

Soit r une relation binaire

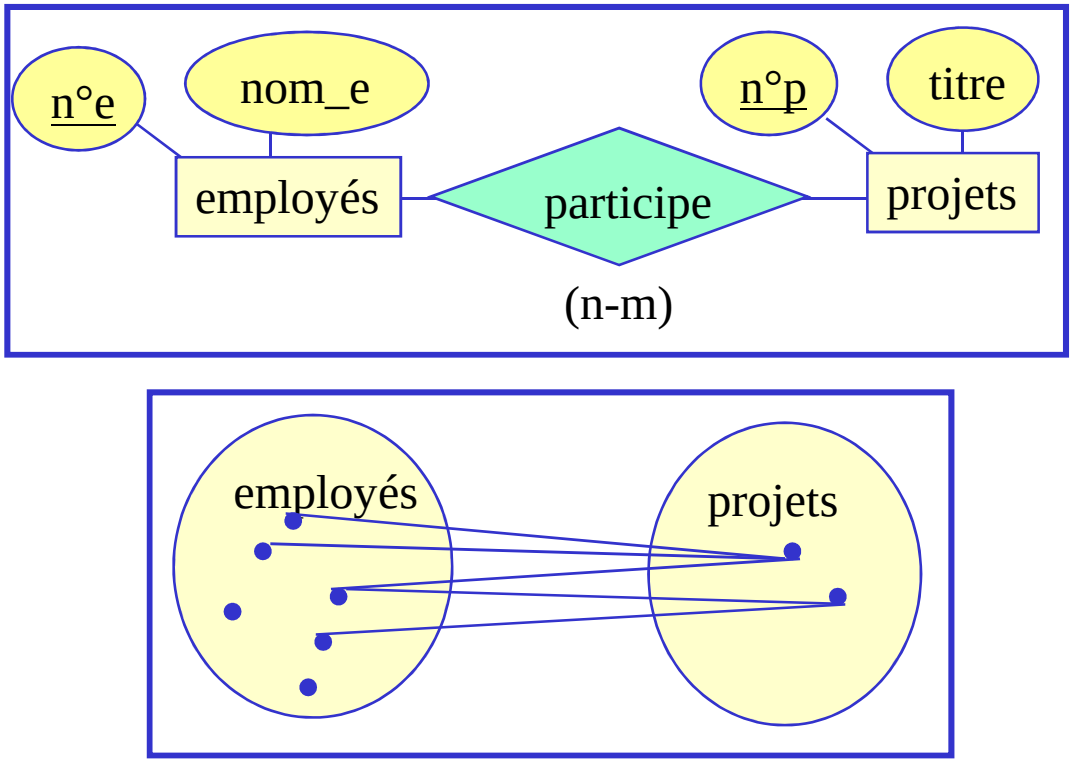
Le type de r est lié au nombre d'occurrences d'une entité qui peuvent être associées avec une occurrence de l'autre entité



Relations de type m-n

Dans une relation "plusieurs à plusieurs" ("many-many", m-n),

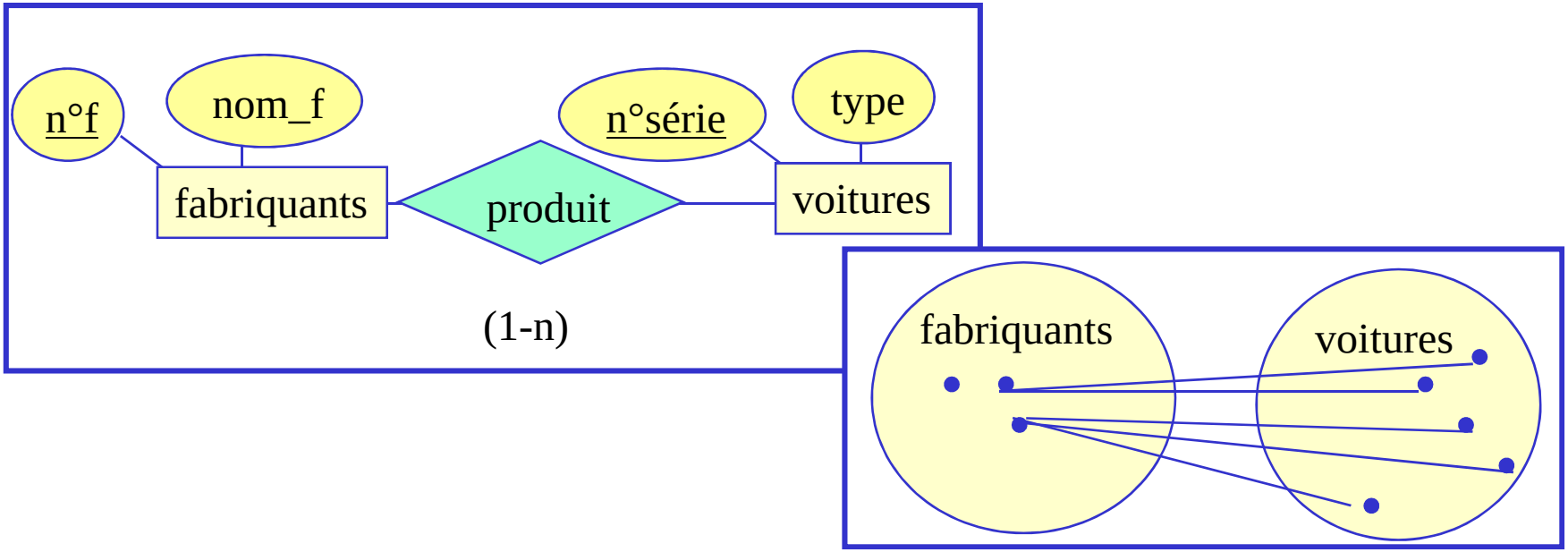
une entité de chaque ensemble peut être connectée à plusieurs entités de l'autre ensemble



Relations de type 1-n

Dans une relation "un à plusieurs" ("one-many", 1-n),

- une entité d'un ensemble peut être connectée à au plus une entité du second ensemble
- mais une entité du second ensemble peut ne pas être connecté à aucune entité du premier ensemble, ou être connectée à une ou plusieurs entité de cet ensemble



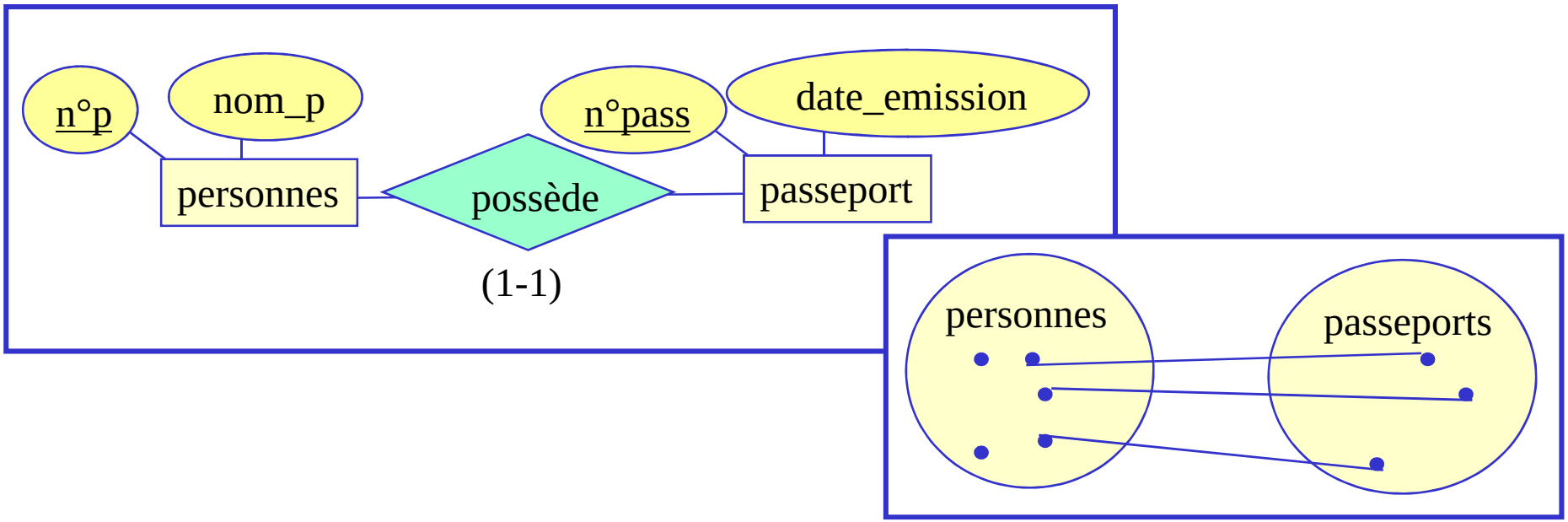
Relations de type 1-1

Dans une relation "un à un" ("one-one", 1-1),

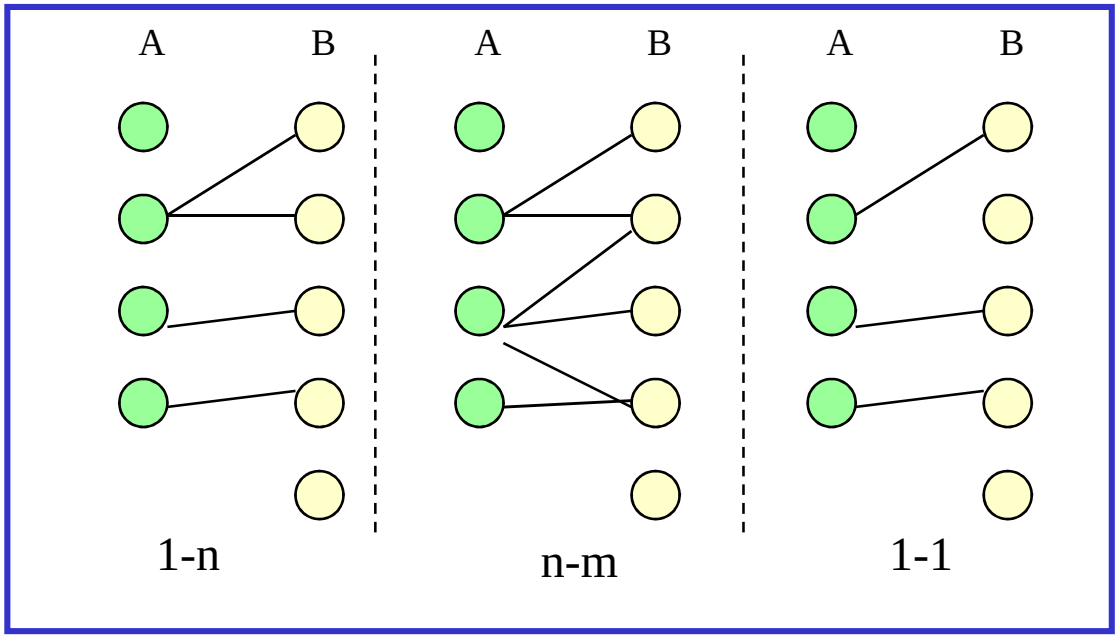
une entité de chaque ensemble peut être connectée à au plus une entité de l'autre ensemble

Exemple: une BD pour l'administration d'un pays

une personne possède au plus un passeport
et un passeport a un seul titulaire

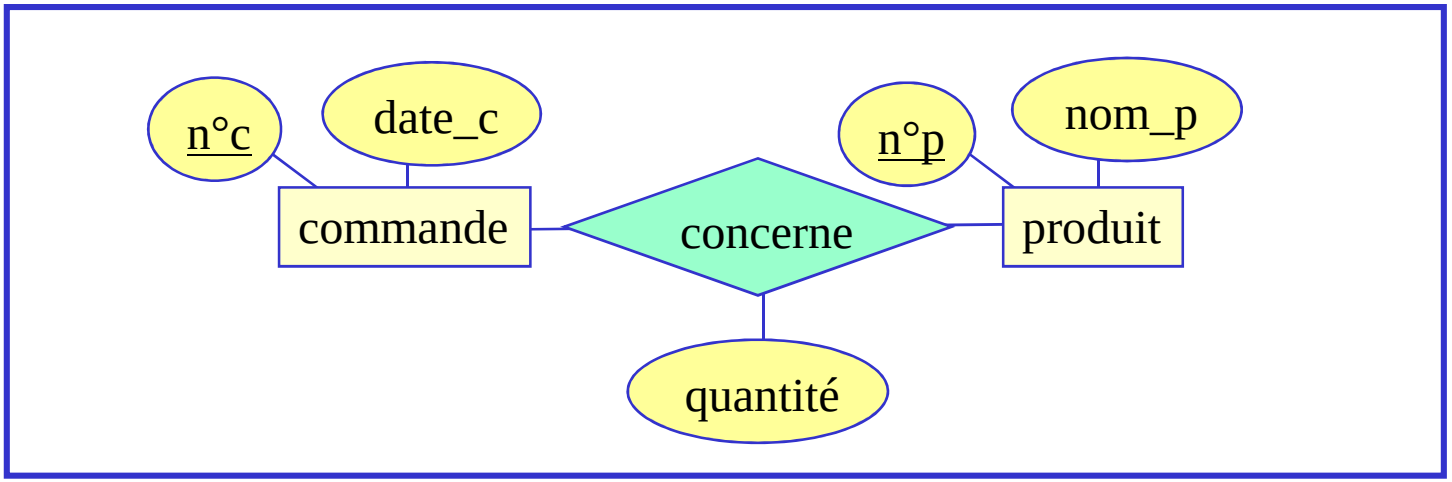


Synthèse sur les types des relations



Attributs de relation

Propriétés dont la valeur dépend des tuples dans l'ensemble de relations.



Cardinalités

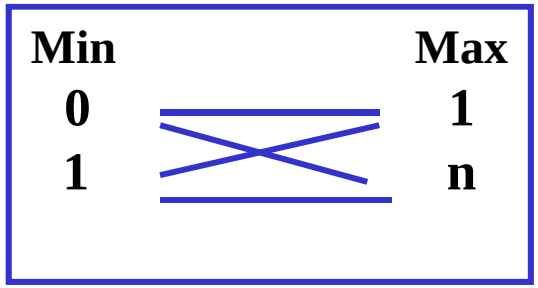
- Les BD sont conçues dans l'hypothèse du "monde fermé"

  Les ensembles d'entités sont FINIS !!

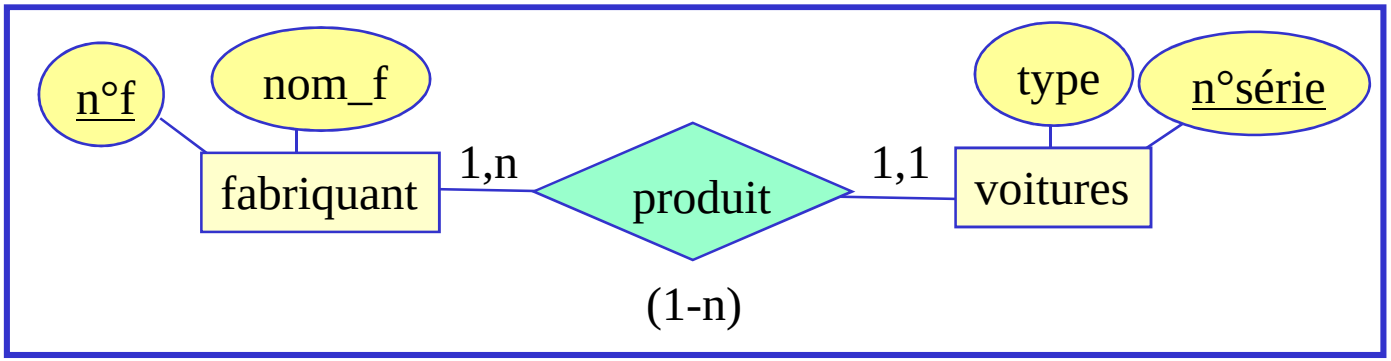
- Cardinalités

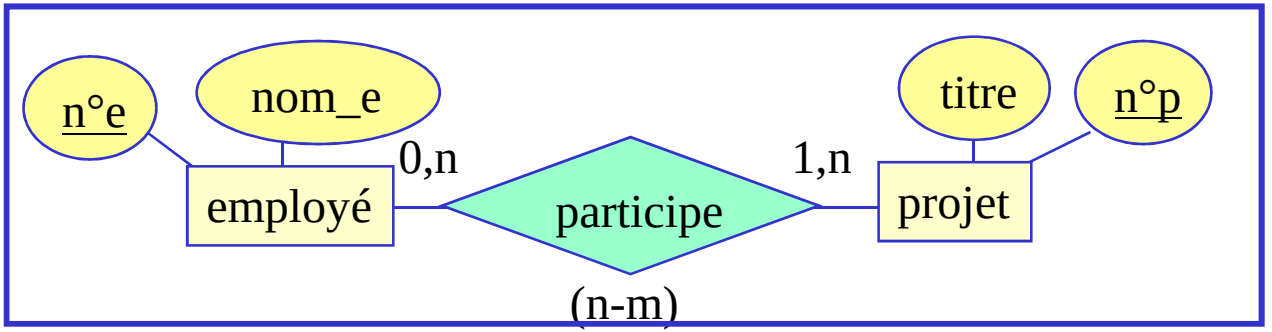
  Couples (min, max) associés à chaque ensemble d'entité relié par une relation r

  - Si r est binaire, entre E_1 et E_2:
    - le min (resp. max) associé à E_i (i=1,2) représente le nombre minimum (resp. max) d'entités de E_j (j=2,1) associées à un élément quelconque de E_i.
  - Si r est n-aire:
    - le nombre d'occurrences possibles d'entités associées dans cette relation quand les autres (n-1) valeurs sont fixées

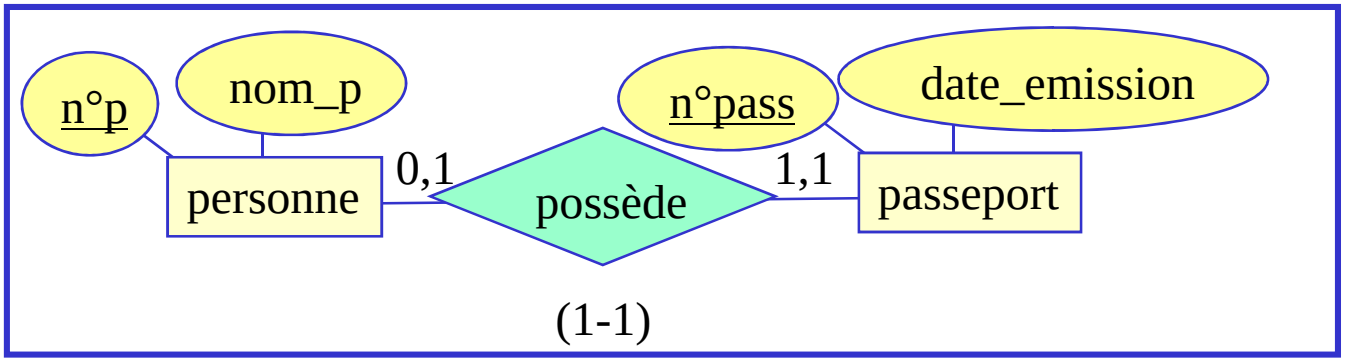


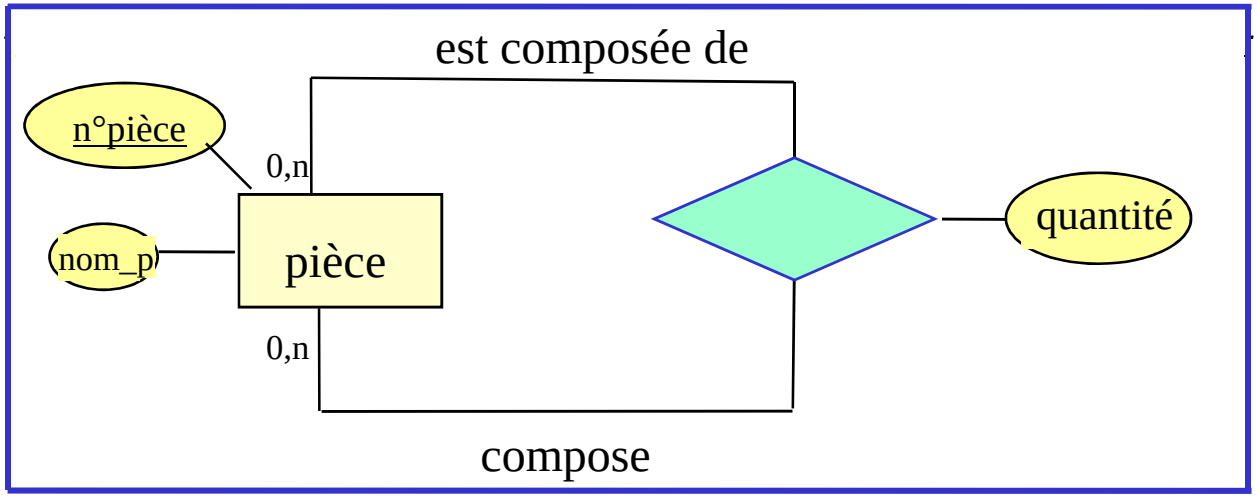
Exemples


n°f
nom_f
fabriquant
1,n
produit
1,1
type
n°série
voitures
(1-n)

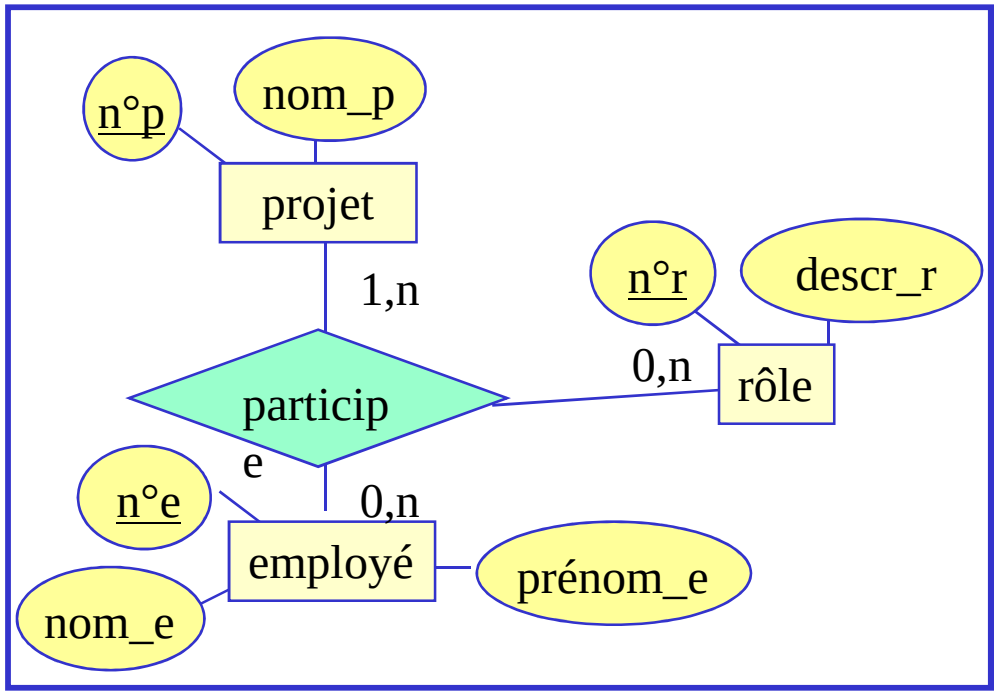

n°e
nom_e
employé
0,n
participe
1,n
titre
n°p
projet
(n-m)

Exemples


n°p
nom_p
personne
0,1
possède
1,1
n°pass
date_emission
passeport
(1-1)


est composée de
n°pièce
0,n
nom_p
pièce
quantité
0,n
compose

Exemple

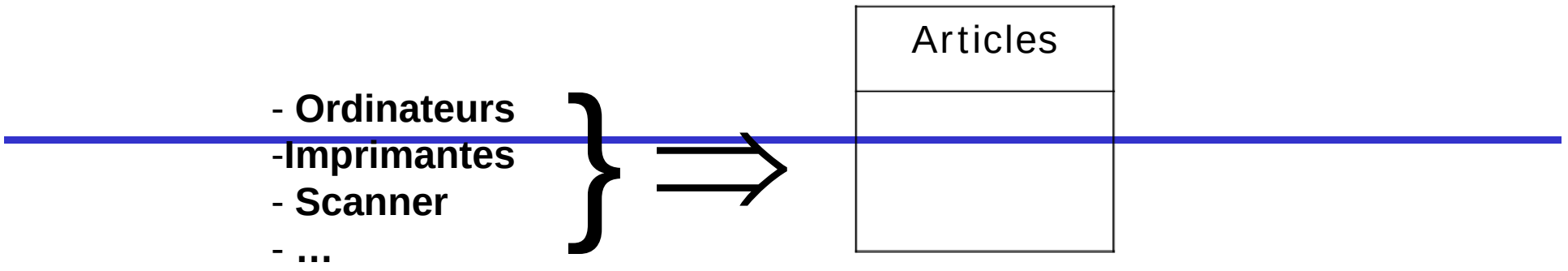

n°p
nom_p
projet
1,n
n°r
descr_r
0,n
rôle
particip
e
n°e
0,n
employé
prénom_e
nom_e

Modèle Entité – Association (synthèse)

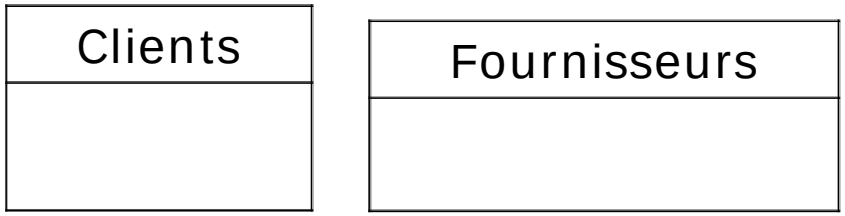
Entité :

Une entité est une population d'individus homogènes.

Exemple: les produits ou les articles vendus par une société peuvent être regroupés dans une même entité Articles.



ceci est possible du fait que ces produits ont les mêmes caractéristiques (par exemple : la désignation, le prix unitaire, la quantité, etc).



Association :

Une association est une liaison qui a une interprétation précise entre plusieurs entités.

Exemple : entre l'entité client et article il y a une liaison qui est Commander : un client commande un article, et entre fournisseurs et clients il y a la liaison : Livrer

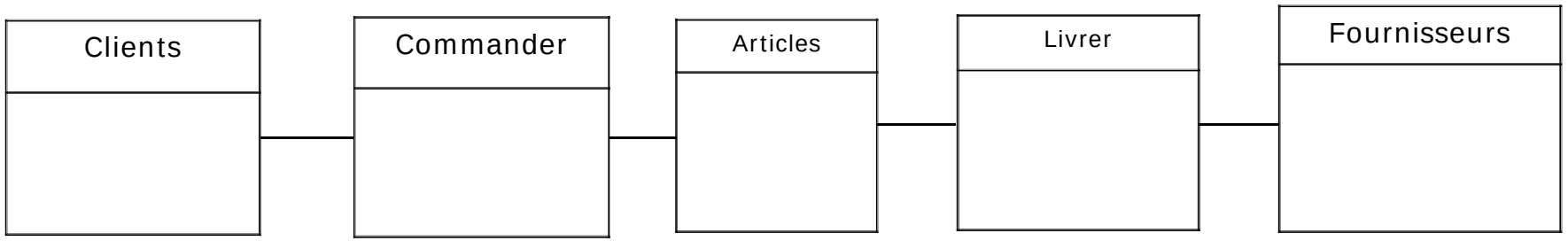


Fig. *Associations*

Attributs et identifiants :

Un attribut est une propriété (caractéristique) d'une entité ou d'une association.

Exemple: Dans l'exemple de la société, l'entité Articles a des attributs que nous avons déjà cités :

- Désignation,
- Quantité,
- prix unitaire,
- ...

L’entité Client peut avoir comme attributs:

- adresse client,
- nom et prénom du client,
- ...

Les associations Commander et Livrer peuvent avoir comme attributs :

- quantité commandée,
- date de livraison,
- ...

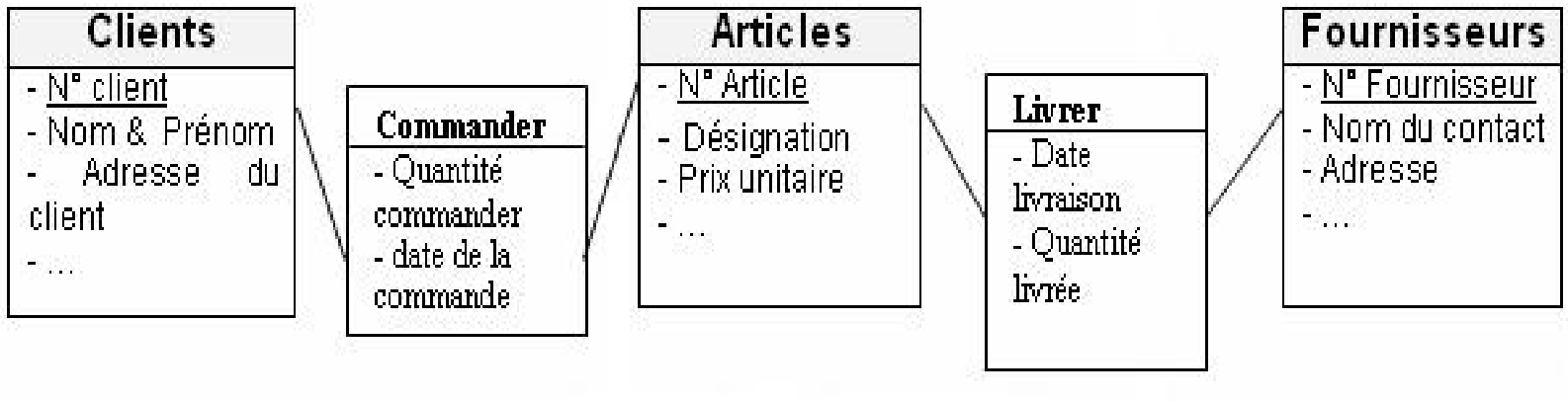


Fig 4. *Attributs et identifiants*

Chaque individu d'une entité doit être identifiable d'une manière unique et sans ambiguïté,

Exemple: *L'*individu **Said Hamidi** de l'entité ***Clients*** ne peut pas être identifier d'une façon unique par son nom :

Plusieurs clients peuvent avoir le même nom

chaque entité doit posséder un attribut sans doublon (ne prenant pas deux fois la même valeur). Il s'agit de

l'identifiant

Remarque : *Une entité doit posséder au moins un attribut qui est son identifiant, par contre une association peut être dépourvu d'attributs.*

Cardinalité

La cardinalité d'un lien entre une entité et une association précise le nombre de fois qu'un individu de l'entité peut être concerné par l'association.

Exemples : un client peut commander de 1 jusqu'à n articles.

Un articles peut être commander 0 fois jusqu'à m fois

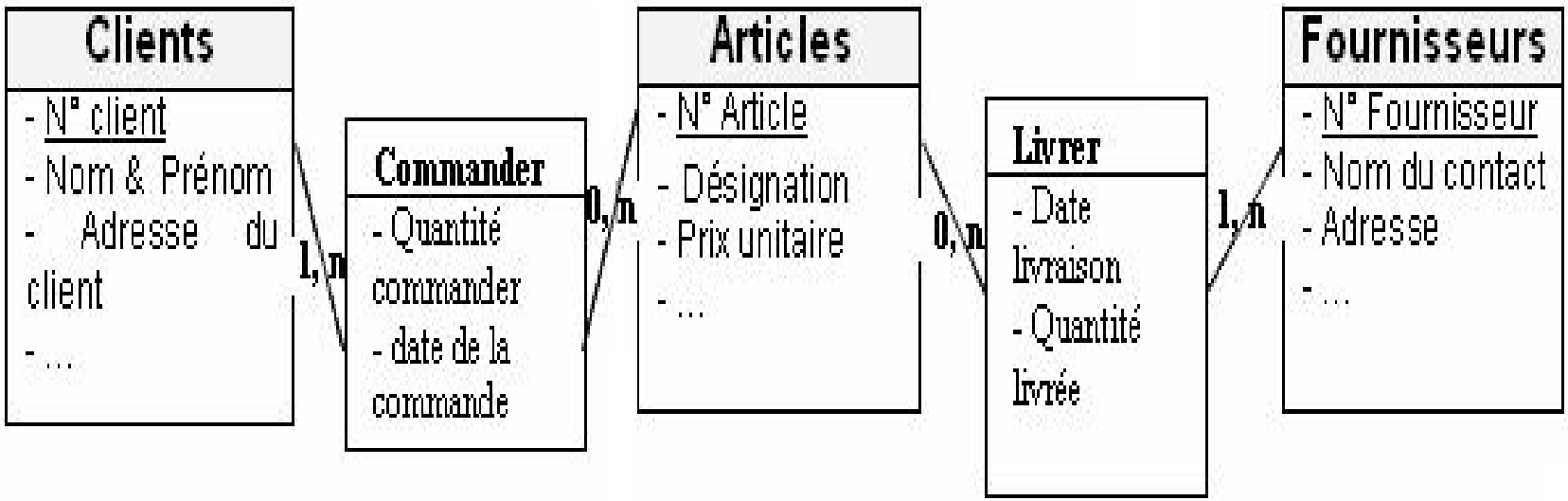


Fig 4. *Cardinalité*

- Un client ne peut exister que s'il commande au moins un article.

Cardinalité minimale pour le client est 1

- Un article peut exister dans le stock même s'il n'est pas commandé par aucun client,

Cardinalité minimale pour l'article est 0.

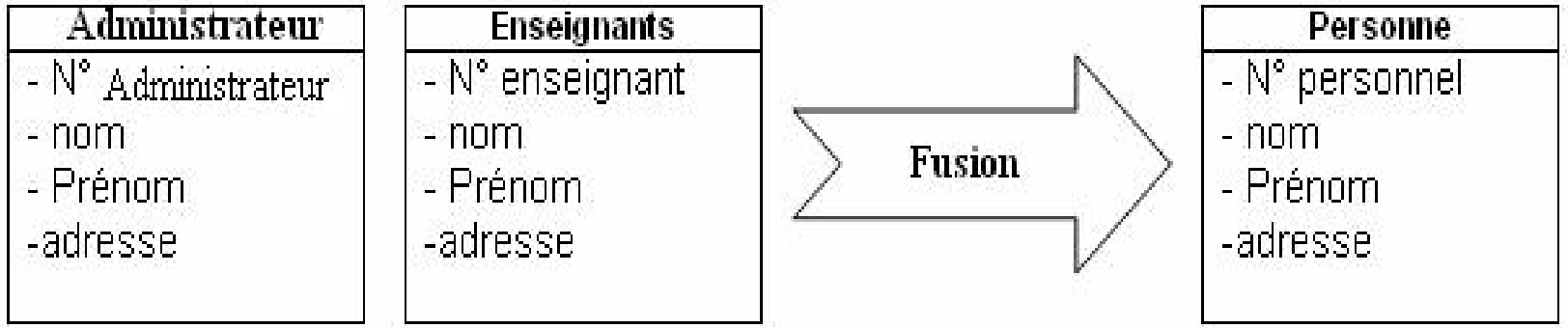
VI.4 Règles de modélisation :

Un bon schéma Entités-association doit vérifier certaines règles dites

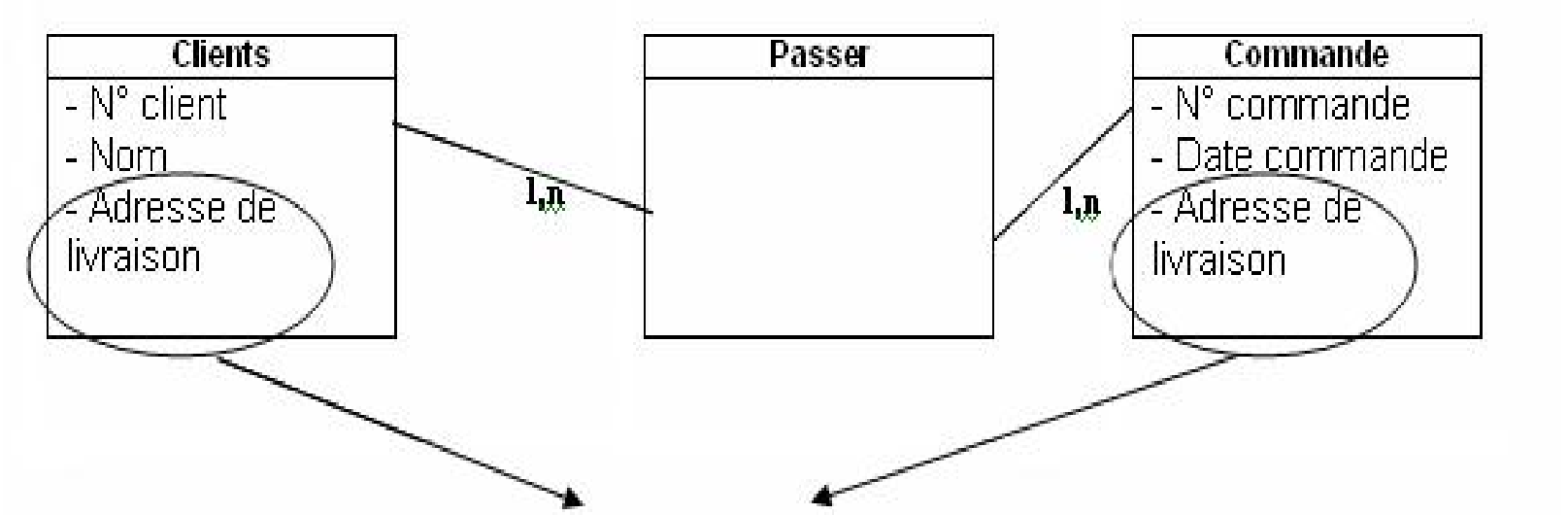
règles de modélisation (normalisation)

Objet : Rassembler les données homogènes et éviter les redondances.

Règ 1: Normalisation des entités : Toutes les entités qui sont remplaçables par une association doivent être remplacées.



Deux entités homogènes peuvent être fusionner



Redondance, donc risque d'incohérence

Les adresses peuvent ne pas être les même donc où va-t-on livrer ?

Règ 2: Normalisation des noms : le nom d'une entité, d'une association ou d'un attribut doit être unique.

Règ 3: Normalisation d'un identifiant : Chaque entité doit posséder un identifiant.

Règ 4: Normalisation des attributs et des associations: les attributs d'une association doivent dépendre directement des identifiants de toutes les entités en association et il faut éliminer les association superflues.

Règ 5. : Normalisation des cardinalités : une cardinalité minimale est toujours 0 ou 1 (pas 2, 3 ou n) et une cardinalité maximale est toujours 1 ou n (pas 2, 3,...).

Conception d'un MCD à partir d'un problème Réel

1- Identifier les entités en présence.

2- Lister leurs attributs.

3- Ajouter les identifiants.

4- Établir les associations entre les entités.

5- Lister leurs attributs.

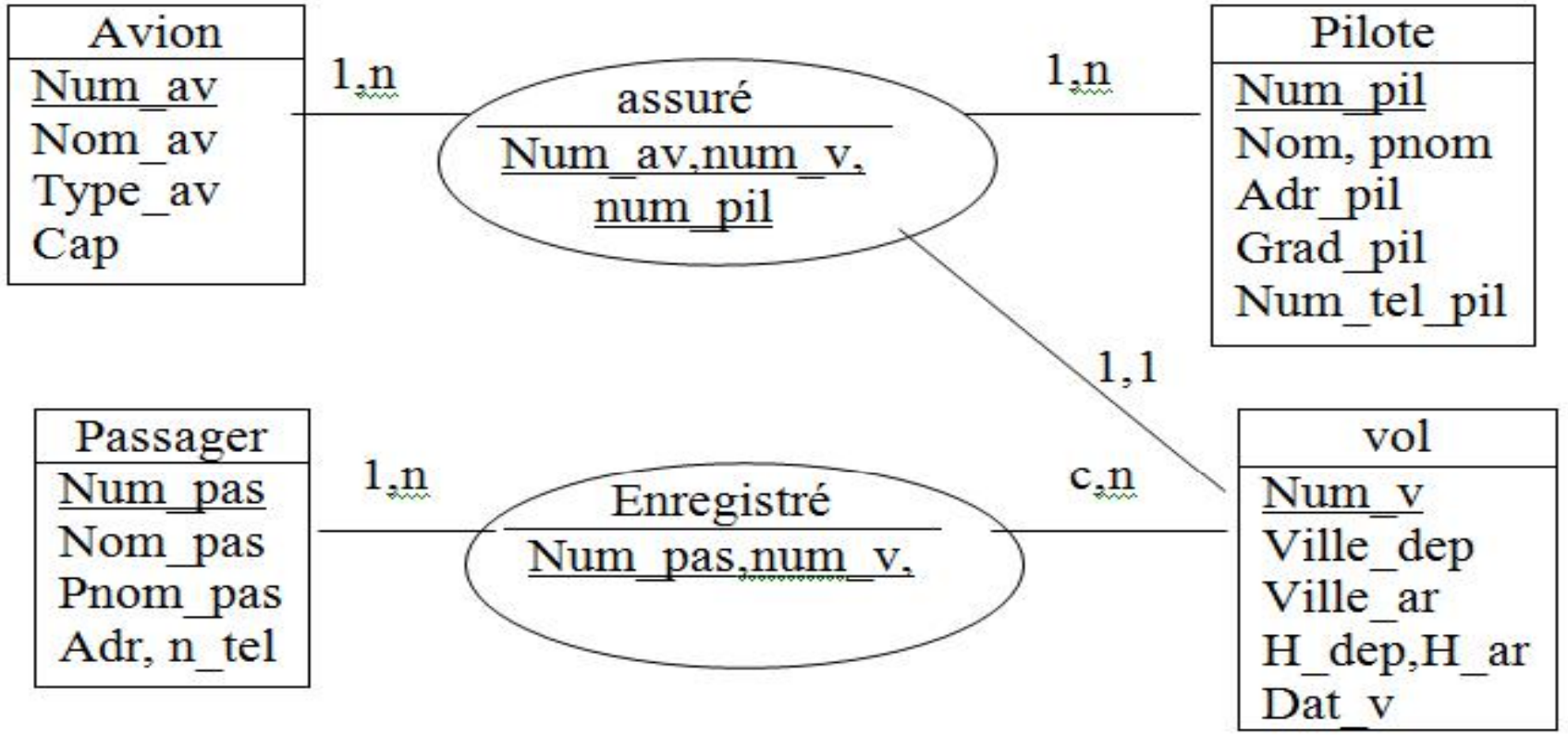
6- Calculer les cardinalités.

7- Vérifier les règles de normalisation.

Exemple 1 (1)

- Exemple: schéma conceptuel : cas d'une compagnie aérienne :
- Soit une compagnie aérienne décrite par les éléments suivants :
- Elle possède un parc d'avion.
- Des pilotes qui assurent un certain nombre de vols.
- Les avions sont identifiés par un numéro et sont caractérisés par un nom, un type et une capacité.
- Les pilotes sont identifiés par un numéro et sont caractérisés par un nom et un prénom, une adresse, un grade et un numéro de téléphone.
- Un vol est assuré par un avion et un pilote.
- Un vol est identifié par un numéro et caractérisé par une ville de départ, une ville d'arrivée, une heure de départ, une heure d'arrivée, et une date.
- Des passagers sont enregistrés sur un vol
- Un passager est identifié par un numéro et caractérisé par un nom et un prénom, une adresse et un numéro de téléphone.

Exemple 1 (2)

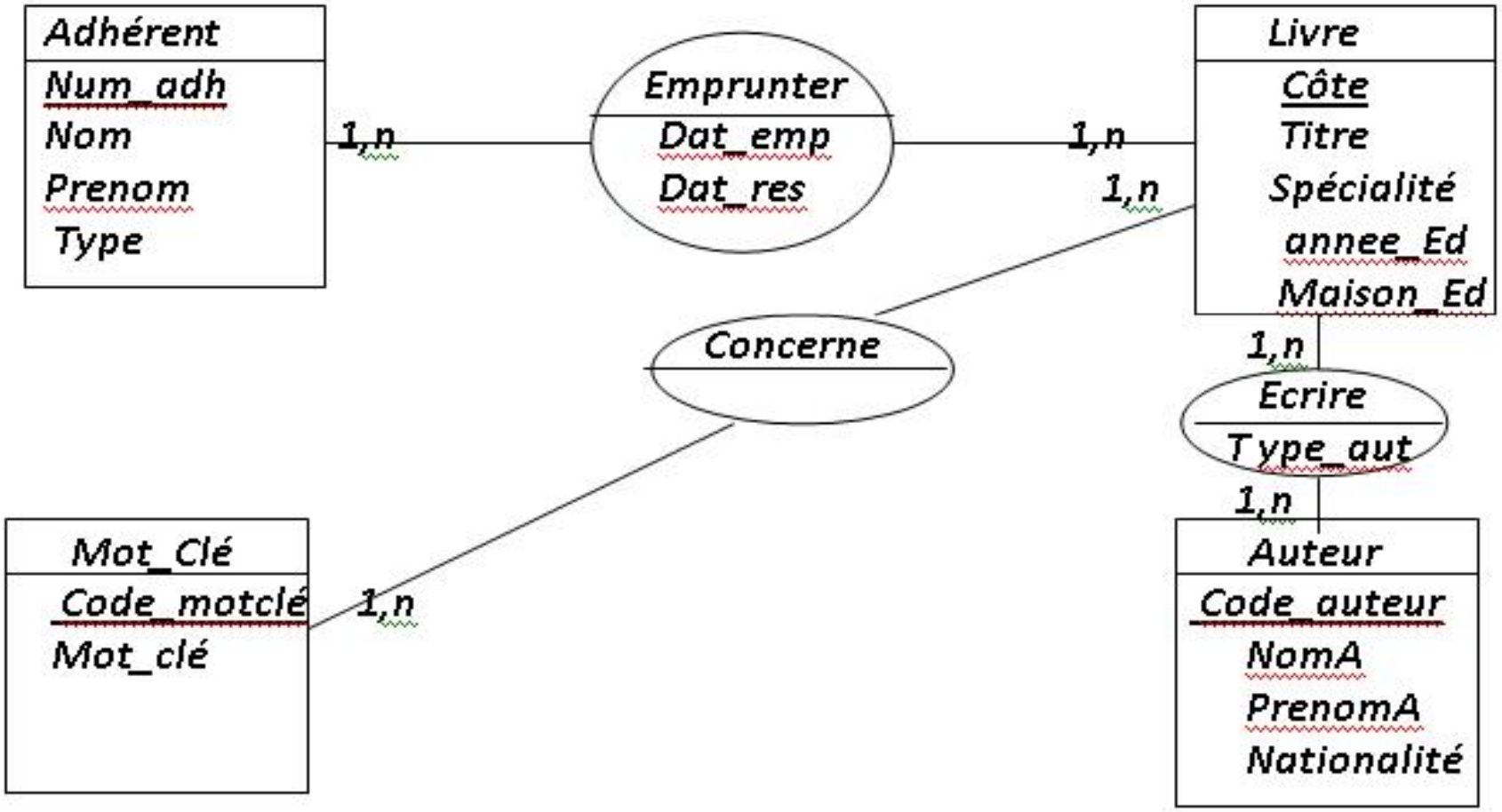


***Remarque*: pour simplifier ce schéma on peut décomposer la relation *assuré* en utilisant les cardinalités individuelles en deux relations : soit *assure1* et *assure2* leur collection sont (avion, vol) et (pilote, vol) respectivement**

Exemple 2 (1)

- Notre établissement universitaire dispose d'une bibliothèque permettant à ses adhérents d'emprunter des livres.
- Un adhérent peut être un étudiant ou un enseignant, il est identifié par un numéro, possède un nom, un prénom et appartient à une catégorie.
- Un livre est identifié par un numéro de référence, possède un titre, une spécialité, une année d'édition et une maison d'édition.
- Un livre est écrit par un ou plusieurs auteurs dont un est principal.
- Un auteur est identifié par un numéro, un nom et un prénom et possède une nationalité.
- Un livre possède un ensemble de mots clés, un mot clé appartient à plusieurs livres et est identifié par un code.
- Un livre peut être emprunté par plusieurs adhérents dans le temps, à chaque opération d'emprunt on doit enregistrer la date et à chaque restitution on doit enregistrer la date.

Exemple 2 (2)


Adhérent
Num_adh
Nom
Prenom
Type
1,n
Emprunter
Dat_emp
Dat_res
1,n
1,n
Livre
Côte
Titre
Spécialité
annee_Ed
Maison_Ed
Concerne
1,n
Ecrire
Type_aut
1,n
Auteur
Code_auteur
NomA
PrenomA
Nationalité
Mot_Clé
Code_motclé
Mot_clé
1,n

Le modèle relationnel

- Introduction :
- introduit au début des années 70, c'est le premier modèle de bases de données indépendant des critères de stockage,
- percevoir la base de données comme un ensemble de relations qui peuvent être représentées sous forme de table à deux dimensions : Les colonnes correspondent aux attributs d'une relation et les lignes correspondent aux tuples.
- utilise un seul concept qui est la relation, ce qui lui rend plus facile à utiliser même pour les utilisateurs ayant plus ou moins de connaissance en informatique.
- Ce modèle est basé essentiellement sur une théorie mathématique « la théorie des ensembles »

Définitions (1)

- Domaine :
  - Un domaine est un ensemble de valeurs caractérisé par un nom.
  - Exemple:
    - Domaine des booléens D1={0,1}
    - Domaine des couleurs D2= {Blanche, Rouge, noire}
    - Domaines des marques de voitures D3={Peugeot, Fiat, Renault}
    - Domaines des modules D4={BD, RO, SI, SE, ANG}

Définitions (2)

- Le produit cartésien :
- Le produit cartésien d'un ensemble de domaines D1, D2, ...,Dn noté D1 × D2 ×.....× Dn est l'ensemble des n-uplets «tuples » (U1, U2, ..., Un) tel que Ui ∈ Di pour tout i=1, 2, ... ,n
- Exemple: En prenant les domaines D2 et D3, le produit cartésien donne 9 couples :

D2	Blanche	Blanche	Blanche	Rouge	Rouge	Rouge	Noire	Noire	Noire
D3	Peugeot	Fiat	Renault	Peugeot	Fiat	Renault	Peugeot	Fiat	Renault

Définitions (3)

- La relation :
- Une relation est un sous-ensemble de produit cartésien d'un ensemble de domaine, une relation est désignée par un nom
- Exemple: Soient les domaines suivants:
  - D1={1, 2, 3}
  - D2= {janvier, février}
  - D3={2002, 2003, 2004}
- On peut avoir une relation date définie sur les domaines D1, D2, et D3 avec les tuples suivants :

D1	D2	D3
1	Janvier	2004
2	Janvier	2004
2	Février	2003
3	Février	2002

Définitions (4)

- L'attribut :
- Un attribut est une colonne de la relation identifié par un nom, c'est une variable qui prend valeur dans un des domaines sur lesquels la relation est définie.
- Exemple: Dans l'exemple précédent on peut avoir les attributs : jour défini sur le domaine D1, mois défini sur le domaine D2 et année défini sur le domaine D3, d'où la relation date devient :

Jour	Mois	Année
1	Janvier	2004
2	Janvier	2004
2	Février	2003
3	Février	2002

Définitions (5)

- Le tuple:
- Un tuple d'une relation désigne une ligne dans la table représentant cette relation.
- Exemple: Dans la relation date (1, janvier, 2005) est un tuple
- 6. Arité d'une relation :
- L'arité d'une relation est le nombre de ses attributs, c'est aussi le nombre de colonnes de la table si on représente la relation sous forme de table.
- Exemple: L'arité de la relation date est égale à 3.

Définitions (6)

- Cardinalité d'une relation :
- La cardinalité d'une relation c'est le nombre de tuples de cette relation, c'est aussi le nombre de lignes de la table qui représente la relation. La cardinalité d'une relation R est un nombre entier noté par |R|
- Exemple: la cardinalité de la relation date : |date| = 4;
- Schéma d'une relation :
- Un schéma d'une relation est composé du nom de la relation suivit de la liste des attributs avec leurs domaines,
- R(A1 : D1, A2 : D2, ..., An : Dn) ;
- Exemple: le schéma de la relation Etudiant :
- Etudiant(numinsc: integer; nom: char(20); prenom: char(20); adr: char(40) ; depart: dep); où : dep : est le domaine des départements {inf, mat, bio, elec} ;

Définitions (7)

- Extension de la relation :
- Une extension de la relation est un ensemble de tuples de la relation, elle est représentée par un tableau à 2 dimensions où une ligne correspond à un tuple et une colonne à un attribut de la relation
- Exemple: Soit la relation produit, une extension de cette relation peut être :

Cod_prod	Désignation	Prix_unit
A002	Armoire	5400
C004	Chaise	1600
E001	Etager	2500

Définitions (8)

- Clé de la relation :
- La clé d'une relation R est un attribut ou un ensemble d'attributs qui permet d'identifier d'une manière unique chaque tuple de la relation.
- Remarque: D'après la définition même d'une relation qu'elle est un ensemble de tuples n'ayant pas d'éléments en double (ne peut pas contenir deux tuples identiques), toute relation possède au moins une clé à savoir : l'ensemble de ses attributs.
- Clé candidate: un attribut ou un ensemble d'attributs est appelé clé candidate s'il peut jouer le rôle de clé dans une relation, une clé doit être soulignée dans un schéma de relation.
- Clé primaire: lorsqu'on dispose pour une relation donnée de plusieurs clés candidates, il est nécessaire de ne retenir qu'une seule parmi l'ensemble des clés candidates et c'est celle qu'on utilisera effectivement pour repérer de manière unique les tuples de la relation, la clé retenue s'appellera alors clé primaire.

Définitions (9)

- Le choix de la clé primaire est généralement effectué en fonction des deux critères suivants :
  - On choisit la clé candidate ayant le plus petit nombre d'attributs : il est préférable de minimiser le nombre d'attributs manipulés.
  - On choisit la clé candidate en fonction de son usage pour la localisation des tuples : il s'agit de privilégier la clé candidate dont l'usage serait le plus fréquent pour localiser les tuples de la relation
- Remarque: toute clé candidate qui n'a pas été retenue comme clé primaire constitue une clé secondaire.

Définitions (10)

- Base de données relationnelles:
- C'est une base de données dont le schéma conceptuel est un ensemble de schémas de relations et dont les occurrences sont des tuples de ces relations.
- Exemple:
- Etudiant(num_insc, nom, prénom, adr, département)
- Module(codmod, intitule, coef)
- Inscrit(num_insc, codmod)
- Examen(num_ins, codmod, note)

Le passage d'un schéma conceptuel E-A vers une base de données relationnelle

- Avec le modèle relationnel, le schéma conceptuel de la base de données est représenté sous forme de table. Une telle représentation ne permet pas de distinguer facilement une entité d'une relation, ni de servir de support de communication si on a à discuter le schéma conceptuel avec d'autres personnes. Généralement on préfère utiliser un autre formalisme tel que celui du modèle entité-association, ce dernier nous offre un aide au niveau de la phase de modélisation même si cela demande un travail supplémentaire de transformation de ce modèle vers le modèle relationnel.
- Pour effectuer ce passage on applique des règles pour les individus et les relations.

Les règles de passage

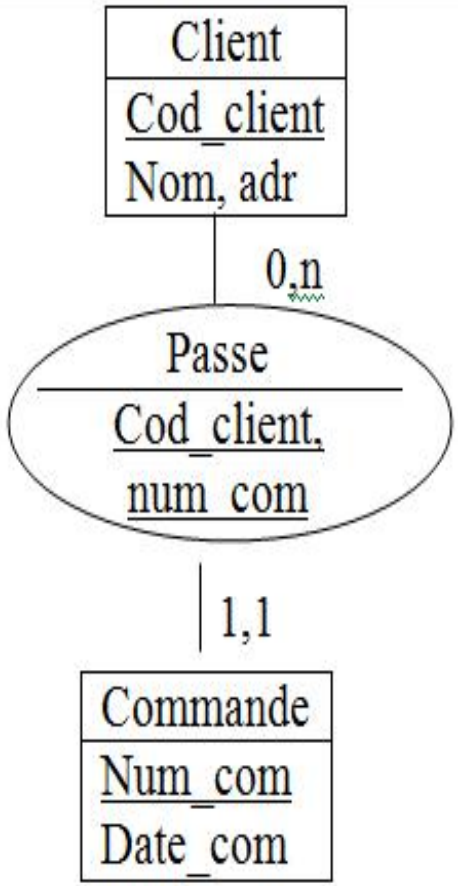
- Pour les individus :
  - Chaque individu se transforme en une table.
  - L'identifiant de l'individu devient la clé primaire de la table.
  - Les propriétés de l'individu deviennent des attributs dans la table.
- Pour les relations :
- Cas des relations père-fils (x, n – y, 1) :
- L'individu père devient la table père.
- L'individu fils devient la table fils.
- L'identifient de l'individu père devient un attribut dans la table fils.
- Les propriétés de la relation deviennent des attributs dans la table fils.

Les règles de passage

- Cas des autres relations :
  - Chaque individu devient une table.
  - L'identifiant de l'individu devient la clé primaire de la table.
  - La relation devient une table.
  - L'identifiant de la relation devient la clé primaire dans la table.
  - Les propriétés de la relation deviennent des attributs dans la table.

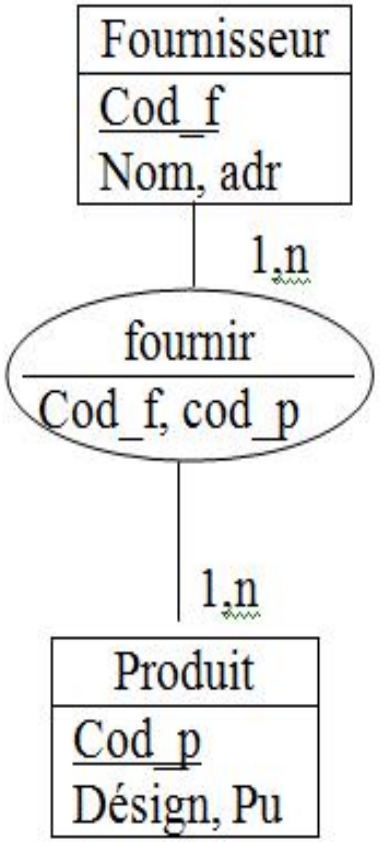
Exemples (1)

1er cas :

Schéma conceptuel E-A	Base de données relationnelle
	Client (cod_client, nom, adr) Commande(num_com, date_com, cod_client)

Exemples (2)

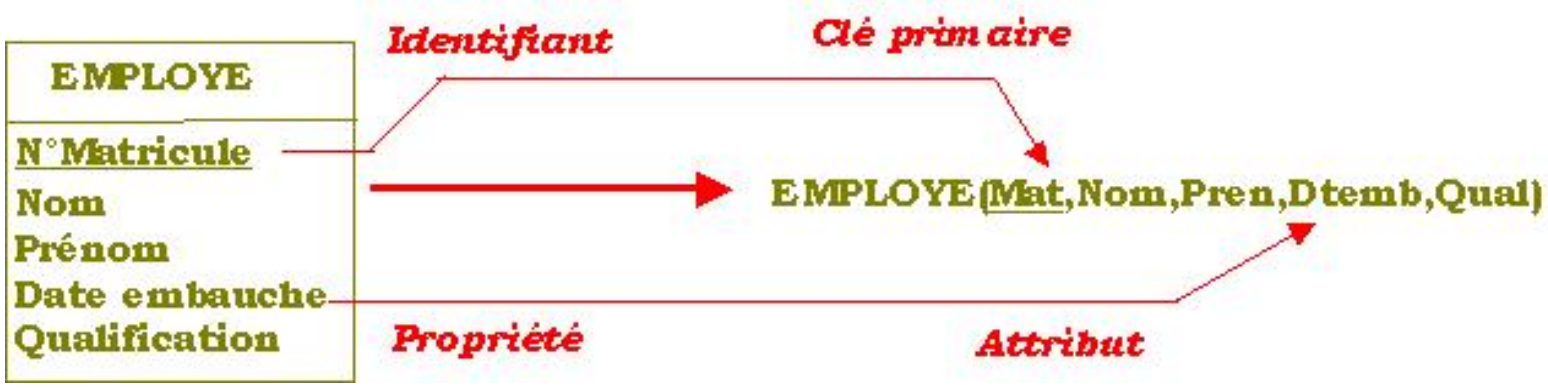
2ème cas :

Schéma conceptuel E-A	Base de données relationnelle
	Fournisseur (cod_f, nom, adr) Fournir (cod_f, cod_p) Produit (cod_p, désign, Pu)

Questions

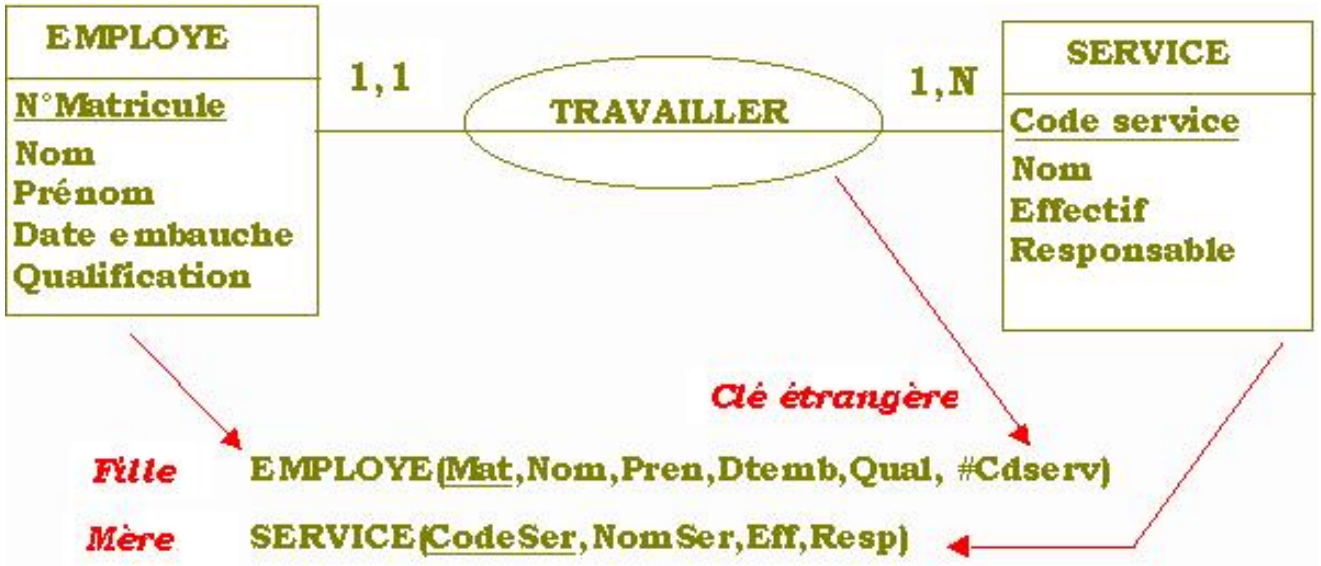
Règles de passage

- REGLE N°1 : TOUTE ENTITE DEVIENT UNE TABLE dans laquelle :
  - les attributs traduisent les propriétés de l'entité
  - la clé primaire traduit l'identifiant de l'entité



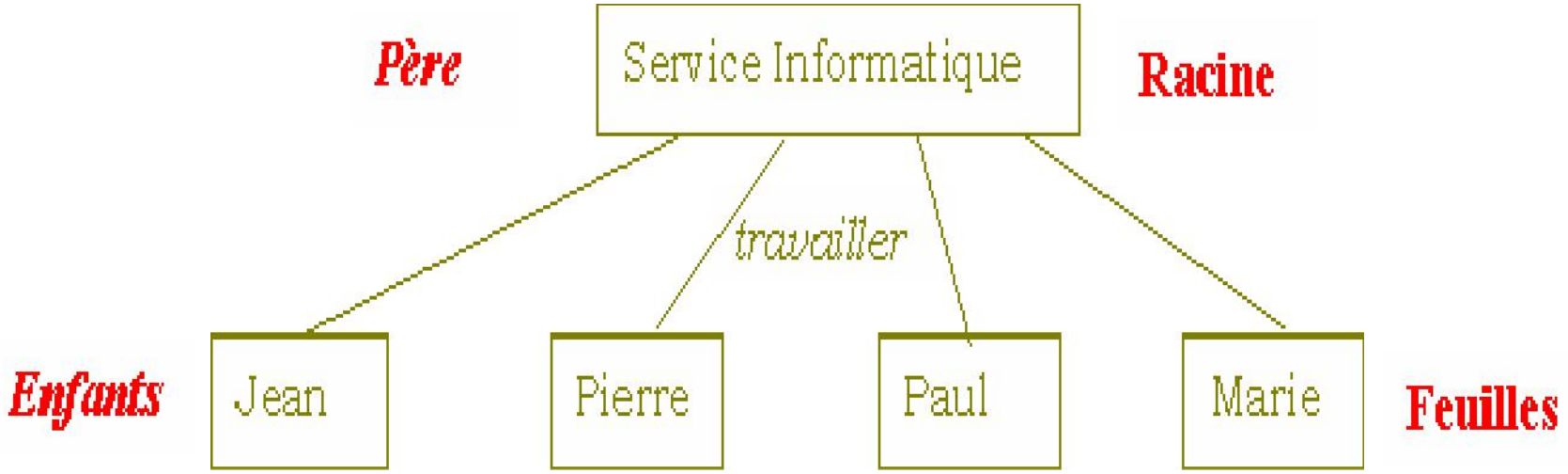
Règles de passage

- REGLE N°2 : UNE ASSOCIATION DE DIMENSION 2 AVEC CARDINALITE 1,1 SE REECRIT EN :
  portant dans la relation fille la clé primaire de la relation mère.
  L'attribut ainsi ajouté s'appelle clé étrangère. Symbole : #.



Règles de passage

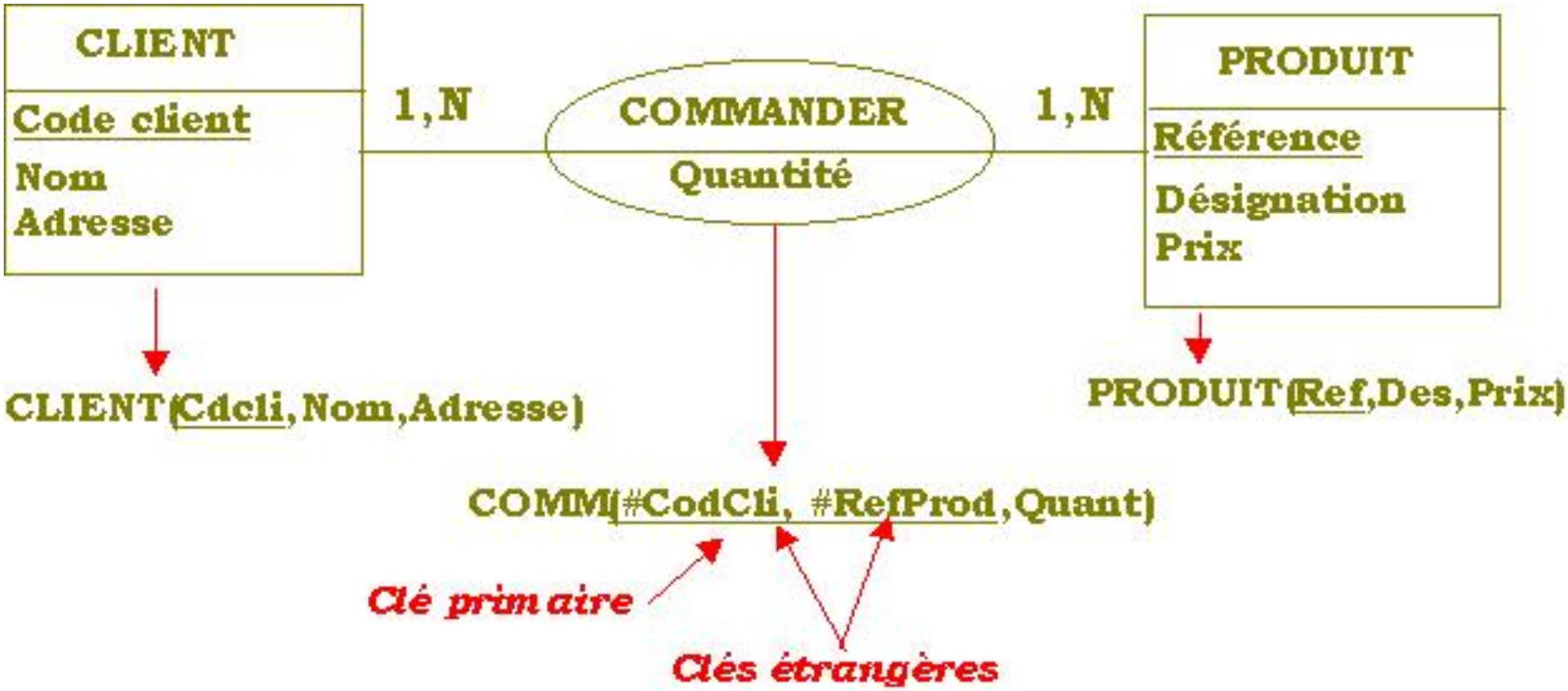
- Relation fille



Règles de passage

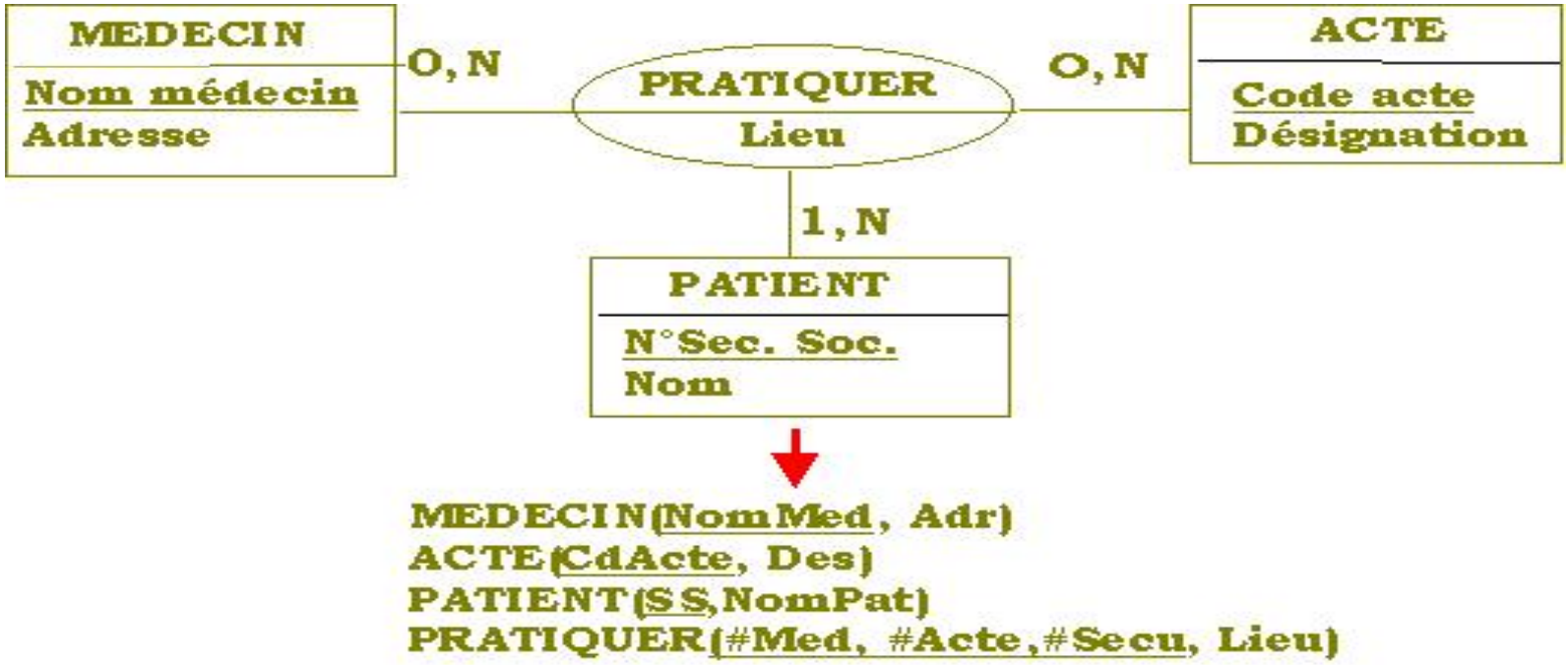
- REGLE N°3 : UNE ASSOCIATION DE DIMENSION 2 AVEC CARDINALITE PLUSIEURS A PLUSIEURS SE REECRIT EN :
  - créant une relation particulière qui contient comme attributs les identifiants des 2 entités associées
  - ces attributs constituent à eux 2 la clé primaire de la relation
    - ils sont individuellement clés étrangères
    - ajoutant la ou les éventuelles propriétés de l'association à cette relation.

Règles de passage



Règles de passage

- REGLE N°4 : UNE ASSOCIATION DE DIMENSION SUPERIEURE A 2 SE REECRIT SELON LA REGLE 3



Règles de passage

- Plus de détail

Chapitre 3

Conception et optimisation de schéma relationnel

La normalisation

- La conception d'un M E-A représente la vision de la réalité de l'analyste. Le formalisme obtenu, établi avec une méthode, ne garantit pas justesse et optimisation.

- La justesse dépend de la compétence et de l'expérience de l'analyste.
- L'optimisation est obtenue par les mécanismes de la normalisation.

Normalisation

- C’est vérifier que la structure devant recevoir des données est organisée pour éviter:
  - des redondances d’informations et
  - des anomalies de conception (sources inévitables d’incohérence à court ou long terme).

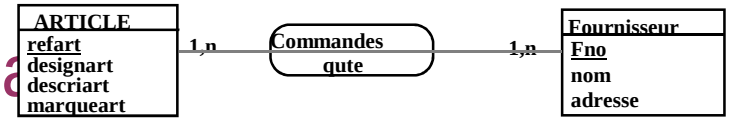
Normalisation

Définition:

Le processus de restructurer le modèle de données logiques pour :

- Eliminer les redondances,
- Organiser les données efficacement,
- Réduire le potentiel d'anomalies pendant les opérations sur les données.

La normalisa



- **Les besoins de normalisation :**
  - **3 problèmes de cohérence lors de mises à jours de la BD:**
    - **anomalie d'insertion (si on rajoute une commande il faut que l'article existe)**
    - **anomalie de suppression (si on enlève un fournisseur il faut enlever toutes les commandes de ce fournisseur)**
    - **anomalie de modification (si on modifie le numéro d'un article il faut modifier toutes les commandes avec ce numéro d'article**

La normalisation

- La théorie de la normalisation repose sur l'analyse de dépendances entre attributs qui sont à l'origine de phénomènes de redondance.
- La normalisation consiste en des méthodes de décomposition des relations.

- Dans certains situations, pour des raisons d'efficacité, on dénormalise.

Normalisation

- Les classifications formelles utilisées pour décrire le niveau de normalisation d'une base de données relationnelle sont appelées les formes normales (FN)

- Il existe 04 formes normales, sont les plus pratiques et sont à connaître.

Normalisation

- Chaque nouvelle forme normale marque une étape supplémentaire de progression vers des relations présentant de moins en moins de redondance

- Le but..

  est d'obtenir une représentation des données présentant un minimum de redondance à l'intérieur de chaque relation et un maximum d'indépendance entre les différentes relations

Formes normales

Première forme normale

Deuxième forme normale

Troisième forme normale

Forme normale de Boyce-Codd

Edgar Codd est à l'origine de la définition des formes normales ...1FN, 2FN, 3FN, BCFN.

la normalisation

- La théorie de la normalisation est basée sur les "dépendances fonctionnelles" (DF).
- C.A.D. Dépendance entre les informations

  Par exemple, le salaire dépend de la qualification

- Les dépendances fonctionnelles traduisent des contraintes sur les données
  - Par exemple, on décide que deux individus différents peuvent avoir même nom et prénom mais jamais le même numéro de sécurité sociale.
  - Ces contraintes sont représentatives d'une perception de la réalité et imposent des limites à la base

Dépendance fonctionnelle

Rappel:

- Définition : dépendance fonctionnelle

  On dit qu'un attribut B dépend fonctionnellement d'un attribut A si, étant donné une valeur de A, il lui correspond une unique valeur de B.

- Notation : A --> B

- Exemple :

  La dépendance fonctionnelle SS → NOM signifie qu'à un numéro est associé un nom seulement.

- Remarquons qu'une dépendance fonctionnelle n'est généralement pas symétrique, c'est-à-dire que SS → NOM n'interdit pas que deux personnes distinctes (correspondant à deux SS différents) portent le même nom.

Propriétés des dépendances fonctionnelles

- Les dépendances fonctionnelles possèdent trois propriétés fondamentales qu'on appelle couramment les axiomes d'Armstrong car mises en évidence par W. Armstrong. En supposant que X, Y , Z, sont trois ensembles d'attributs, il est utile de faire les remarques suivantes avant d'énoncer les propriétés :
- L'écriture X , Z est une écriture simplifiée de l'union ensembliste des deux ensembles d'attributs X et Z. Ceci veut dire que l'écriture X , Z devra être interprétée comme $X \cup Z$.
- Il faut souligner aussi que l'union ensembliste est un opérateur associatif et commutatif :
- $((X \cup Y) \cup Z) = (X \cup (Y \cup Z))$
- $X \cup Z = Z \cup X$

Propriétés des dépendances fonctionnelles (5)

- Réflexivité
  - Si $X \subseteq Y \Rightarrow Y \rightarrow X$
  - Ex : $A \subseteq A,B \Rightarrow A,B \rightarrow A$ et $A,B \rightarrow B$
- Transitivité
  - Si $X \rightarrow Y$ et $Y \rightarrow Z \Rightarrow X \rightarrow Z$
  - Ex : $A,B \rightarrow C$ et $C \rightarrow D \Rightarrow A,B \rightarrow D$
- Augmentation
  - Si $X \rightarrow Y \Rightarrow \forall Z \quad X, Z \rightarrow Y, Z$
  - Ex : $A,B \rightarrow C \Rightarrow A,B,D \rightarrow C,D$

A partir de ces trois axiomes de base, on peut déduire facilement d'autres propriétés très utiles en pratiques. Les plus remarquables sont :

Propriétés des dépendances fonctionnelles (6)

- Union des parties droites
  - Si X → Y et X → Z ⇒ X → Y, Z
  - Ex : A →C et A →D ⇒ A →C,D
- Décomposition
  - Si X → Y, Z ⇒ X → Y et X → Z
  - Ex : A →C,D ⇒ A →C et A →D
- Pseudo Transitivité
  - Si X → Y et Y , W → Z ⇒ X , W → Z
  - Ex : A →C et B,C →D ⇒ A,B → D
- Distributivité par rapport à l'union
  - Si X → U ⇒ ∀ ui ⊆ U X → ui (c'est une généralisation de la décomposition)
  - Ex : A →C,D,E,F ⇒ A → C et A → D et A → E et A → F

    A,B →C,D,E ⇒ A,B → C et A,B → D et A,B → E

Propriétés des dépendances fonctionnelles (7)

Dépendance fonctionnelle élémentaire

- On appelle dépendance fonctionnelle élémentaire une dépendance de la forme : $X \rightarrow A$, ou A est un attribut unique non inclus dans X ($A \not\subset X$), tel que $\forall X' \subset X$, il n'existe pas de dépendance fonctionnelle $X' \rightarrow A$.

- En d'autres termes, une dépendance fonctionnelle est élémentaire si elle n'est pas obtenue par augmentation de sa partie gauche.

Propriétés des dépendances fonctionnelles (8)

- exemple :
- Si on considère les deux D.F. :
- (NUMERO, NOM) → ADRESSE (1)
- NUMERO → ADRESSE (2)
- la D.F. (1) est une D.F. non élémentaire car NUMERO ⊂ (NUMERO,NOM) et il existe la D.F. NUMERO → ADRESSE.
- Par contre la D.F. (2) est une D.F. élémentaire car elle vérifie les propriétés d'une D.F. élémentaire

Propriétés des dépendances fonctionnelles (9)

- Dépendance fonctionnelle transitive
- On appelle dépendance fonctionnelle transitive une dépendance de la forme : $X \rightarrow A$, ou A est un attribut unique non inclus dans X ($A \not\subset X$), tel que :

  $\exists\ Y \not\subset X, X \rightarrow Y$ et $Y \rightarrow A$ et $Y \not\rightarrow X$
- Dans ce cas, on dit aussi que A est transitivement dépendant de X. Dans le cas où A n'est pas
- transitivement dépendant de X, on dit que A est directement dépendant de X

Propriétés des dépendances fonctionnelles (10)

- exemple :
- Si on considère les D.F. :
- 1 - A → B 2- B → C 3- A → C 4- D → B
  5- D → C 6 B → D
- La D.F. A → C est transitive car ∃ B ⊄ A tel que :
- A → B et B → C et B /→ A
- Par contre la D.F. D →C n'est pas transitive puisque ∃ B ⊄ D tel que :
- D → B et B → C mais B → D

Propriétés des dépendances fonctionnelles (12)

- Dépendance fonctionnelle directe

On appelle dépendance fonctionnelle directe une dépendance de la forme $X \rightarrow A$, ou A est un attribut unique non inclus dans X ($A \not\subset X$), tel que :

- $\forall Y \not\subset X$, si $X \rightarrow Y$ et $Y \rightarrow A$ alors la d.f. $Y \rightarrow X$ existe aussi

Propriétés des dépendances fonctionnelles (13)

- exemple :
- Si on considère les D.F. :
- 1- A → B 2-B → C 3- A → C 4-D → B 5-D → C 6-B → D
- La D.F. A → C n'est pas directe car pour B ⊄ A on a : A → B et B → C = / = > B → A
- La D.F. D →C est directe puisque pour B ⊄ D on a : D → B et B → C ⇒ B → D existe

Propriétés des dépendances fonctionnelles (14)

- La D.F. A $\rightarrow$B est directe puisque il n'existe pas d'attribut Y $\notin$ A tel qu'on a en même temps les d.f. A $\rightarrow$Y et Y $\rightarrow$B pour vérifier s'il y a ou non contradiction avec la définition. Dans ces cas là , la d.f. est forcément directe.

- Dans le cas où A n'est pas directement dépendant de X (i.e. la d.f. Y $\not\rightarrow$ X n'existe pas), on dit que A est transitivement dépendant de X.

Propriétés des dépendances fonctionnelles (15)

Dépendance fonctionnelle totale, pleine ou complète

- On appelle dépendance fonctionnelle totale , pleine ou complète une dépendance fonctionnelle de la forme $X \rightarrow A$, ou A est un attribut unique non inclus dans X ($A \notin X$), telle que $\forall X' \subset X$, il n'existe pas de dépendance fonctionnelle $X' \rightarrow A$. En d'autres termes, la dépendance $X \rightarrow A$ est élémentaire.

- Dans ce cas on dit aussi que A est pleinement ou totalement ou complètement dépendant de X.

- Dans le cas contraire, on dit aussi que A est partiellement dépendant de X (i.e. A dépend d'une partie de X et donc forcément de X aussi par augmentation)

Propriétés des dépendances fonctionnelles (16)

- exemple :
- Si on considère les D.F. :

  1- A ,B $\not\rightarrow$ C 2- B $\rightarrow$ C 3- B,C $\rightarrow$ D
- dans la d.f. 1 C n'est pas totalement dépendant de A ,B car on a B $\subset$ A,B tel que B $\rightarrow$ C. On peut dire que C est partiellement dépendant de A,B.
- dans la d.f. 3- D est totalement dépendant de B,C car il n'existe pas de X' $\subset$ B ,C et tel que X' $\rightarrow$ D.

Propriétés des dépendances fonctionnelles (17)

Dépendance fonctionnelle triviale

On appelle dépendance fonctionnelle triviale une dépendance fonctionnelle de la forme $X \rightarrow Y$, tel que $Y \subseteq X$. En d'autres termes, une dépendance fonctionnelle est triviale si elle est obtenue grâce à la propriété de réflexivité.

- exemple : Si on considère les D.F. :

1- A ,B → B 2-B,C → C 3- A ,C→ D

- la d.f. 1 est triviale car on a $B \subset A$,B
- la d.f. 2 est triviale car on a $C \subset B$,C
- la d.f. 3 n'est pas triviale car on a $D \not\subset A$,C

Propriétés des dépendances fonctionnelles (18)

les notions de dépendance fonctionnelle élémentaire, de dépendance fonctionnelle totale, de dépendance fonctionnelle directe vont intervenir dans la définition des formes normales d'une relation et principalement la deuxième et la troisième formes normales.

Normalisation : 1FN

1FN (rubrique élémentaire) :

– Un MR est dit en première forme normale, si toutes les entités sont composées d'attributs élémentaires ou atomiques (hors clefs concaténées).

IP	Role
192.168.0.1	client
192.168.0.2	serveur
192.168.0.4	**client et serveur**

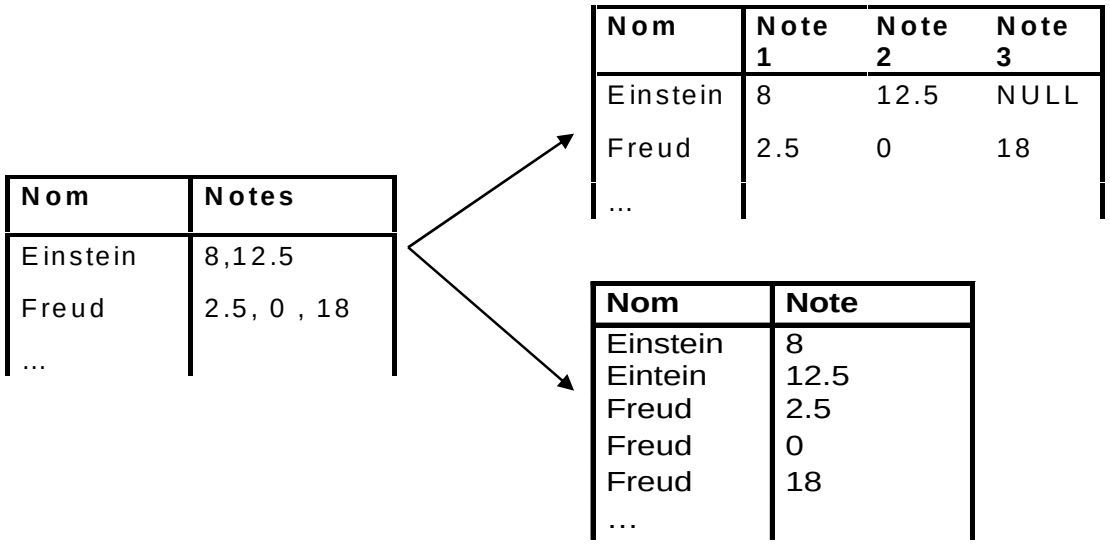
Conséquences :

– un attribut représente une donnée élémentaire du monde réel ;

– Un attribut ne peut désigner, ni une donnée composée d'entités de natures différentes, ni une liste de données de même nature.

Normalisation : 1FN

- Si on veut avoir les notes d'un étudiant :



- N'est pas en 1FN. Pour y remédier, on peut soit créer un attribut par note si le nombre maximal de notes est connu et si le nombre moyen de notes par personne est proche de ce maximum, Soit fabrique autant de tuples que de couples (nom, note)

Normalisation : 2FN

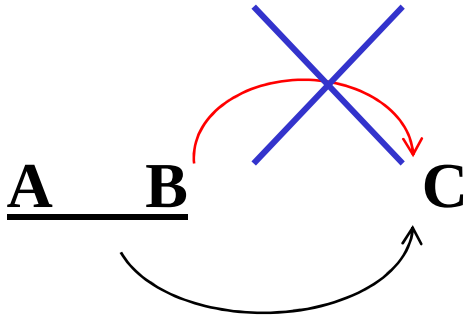
2FN (dépendance fonctionnelle élémentaire) :

– Un MR est dit en deuxième forme normale, si:

- il remplit les conditions de la 1FN et si,
- tout attribut n'appartenant pas à une clé ne dépend pas d'une partie de cette clé.

A,B -> C

B->C



Normalisation : 2FN

- La 2FN est basée sur le concept de dépendance fonctionnelle totale
  - Définition de la dépendance fonctionnelle totale x → y: si la suppression d'un attribut A de x signifie que la dépendance n'existe plus

Normalisation : 2FN

- Exemple:

  Considérons une table « Membres du Departement" dont les attributs sont :

  ID departement,

  ID employe,

  Date de naissance employe;

  et supposons qu'un employé travaille dans un ou plusieurs départements.

La combinaison de ID departement et de ID employe identifie de manière unique un enregistrement de la table.

Est-ce que cette table est en 2FN?

Comme Date de naissance employe ne dépend que d'un seul de ces attributs – l'ID employe – la table n'est pas 2NF.

Normalisation : 2FN

Autre exemple :

- Considérons la relation PLAGE de schéma suivant :
  PLAGE (NOMP, REGION, TYPE, POLLUTION)

où la clé est (NOMP, REGION). Supposons que la pollution est bien dépendante de la plage (identifiée par (NOMP, REGION)) mais que le type est, quant à lui, dépendant de la région seule.

Que pouvons nous faire?:

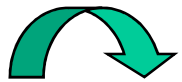
La deuxième forme normale nous impose de distinguer deux relations R1 et R2 de schémas respectifs :

- R1 (NOMP, REGION, POLLUTION) ;
  R2 (REGION, TYPE).

Normalisation : 3FN

Définition : troisième forme normale:

- Une relation est en troisième forme normale si :
  - elle est en deuxième forme normale ;
  - tout attribut n'appartenant pas à une clé ne dépend pas d'un attribut non clé.
  - cad toute DF dont la partie droite n'est pas une clé doit avoir une partie gauche qui est une clé :



B, C **OK**

A, B, C

Normalisation : 3FN

- Exemple :

  Considérons une table "Departements" dont les attributs sont

  ID departement,

  Nom departement,

  ID manager,

  Date embauche manager

  et supposons que chaque manager peut gérer un ou plusieurs départements.

  ID departement est une clé candidate.

  Bien que Date embauche manager est fonctionnellement dépendante de {ID departement}, elle est aussi fonctionnellement dépendante de l'attribut ID manager.

  ID departement → Date embauche manager

  ID manager → Date embauche manager X

Ceci signifie que la table n'est pas 3FN.

Departements (ID department, Nom departement, ID manager

Information Manager (ID manager, date embauche manager)

Normalisation : 3FN

Autre exemple:

- Considérons maintenant la relation PLAGE de schéma

  PLAGE (NP, REGION, TYPE, POLLUTION)

où la clé est NP.

Supposons maintenant comme dans l'exemple précédent que le type est dépendant de la région.

La troisième forme normale nous impose de distinguer deux relations R1 et R2 de schémas respectifs :

R1 (NP, REGION, POLLUTION) ;
R2 (REGION, TYPE).

Normalisation : 3FN

Autre Exemple :

Enseignant (Nom, Bureau, Batiment, Discipline, telephone)

Avec des contraintes d'intégrité : Un bâtiment héberge des enseignants d'une même discipline; un bureau donné possède un numéro de d'appel unique.

Avec les DF: Batiment → Discipline
Batiment, Bureau →Telephone
Telephone → Bureau, Batiment

Note: la deuxième contrainte définit une relation 1-1, donc une DF reflexive.

Enseignant (Nom, Bureau, Batiment, Discipline, telephone)

- La dépendance Batiment->Discipline ne concerne pas la clé. La relation Enseignant n'est pas 3FN.
- On sort donc l'attribut Discipline de la relation Enseignant (dépendant d'un attribut non-clé) et on créé une nouvelle relation Affectation:

  Affectation(Batiment,Discipline)

- Les dépendances

  Batiment, Bureau->Telephone

  Telephone->Bureau, Batiment

  ne concernent pas la clé. On créé la nouvelle relation:

  Annuaire(Bureau, Batiment, telephone)

- La relation Enseignant d'origine peut être renommée Siège, avec les attributs restant:

  Siege(Nom,Bureau,Batiment)

Normalisation : Boyce-Codd

- La FNBC était proposée comme une forme plus simple de 3FN, mais elle est finalement plus stricte que la 3FN.

Normalisation : BCFN

Définition : Boyce-Codd forme normale:

- Une relation est en BCFN si :
  - elle est en 3 eme forme normale ;
  - aucun attribut ne dépend d'un attribut non clé.

  Toute relation qui est en BCFN est forcement en 3 FN
- C.a.d. toute DF a pour partie gauche (origine de la flèche) un clé candidate ou primaire entière.
- Une relation en BCFN si quelle que soit la DF, le membre gauche est une clé.

A, B, C

Normalisation : Exemple FNBC

Considérons la relation suivante: Cours (Matiere, Classe, Professeur)

complétée par les règles de gestion suivantes:

un professeur n'enseigne qu'une seule matière,

un classe n'a qu'un seul enseignant par matière

desquelles on déduit les DF suivantes:

Matière, Classe → Professeur

Professeur → Matière

Cette relation est en 3NF, néanmoins il est impossible d'enregistrer un professeur sans classe affectée (puisque classe est une partie de la clé primaire de la table Cours), et la disparition d'une classe peut entraîner la disparition de professeur.

Ceci est du au fait qu'une DF n'ait pas comme origine une clé de la relation.

Normalisation : Exemple FNBC

On pourrait alors décomposer la table Cours en 2 tables:

Spécialité (Professeur, Matière)

Enseignant (Classe, Professeur)

Mais la première DF serait alors perdue...

Normalisation : 2ième exemple FNBC

Considérons la relation suivante:

Fournisseur (nom, adresse, produit, prix)

avec les DF suivantes :

DF1: nom → adresse

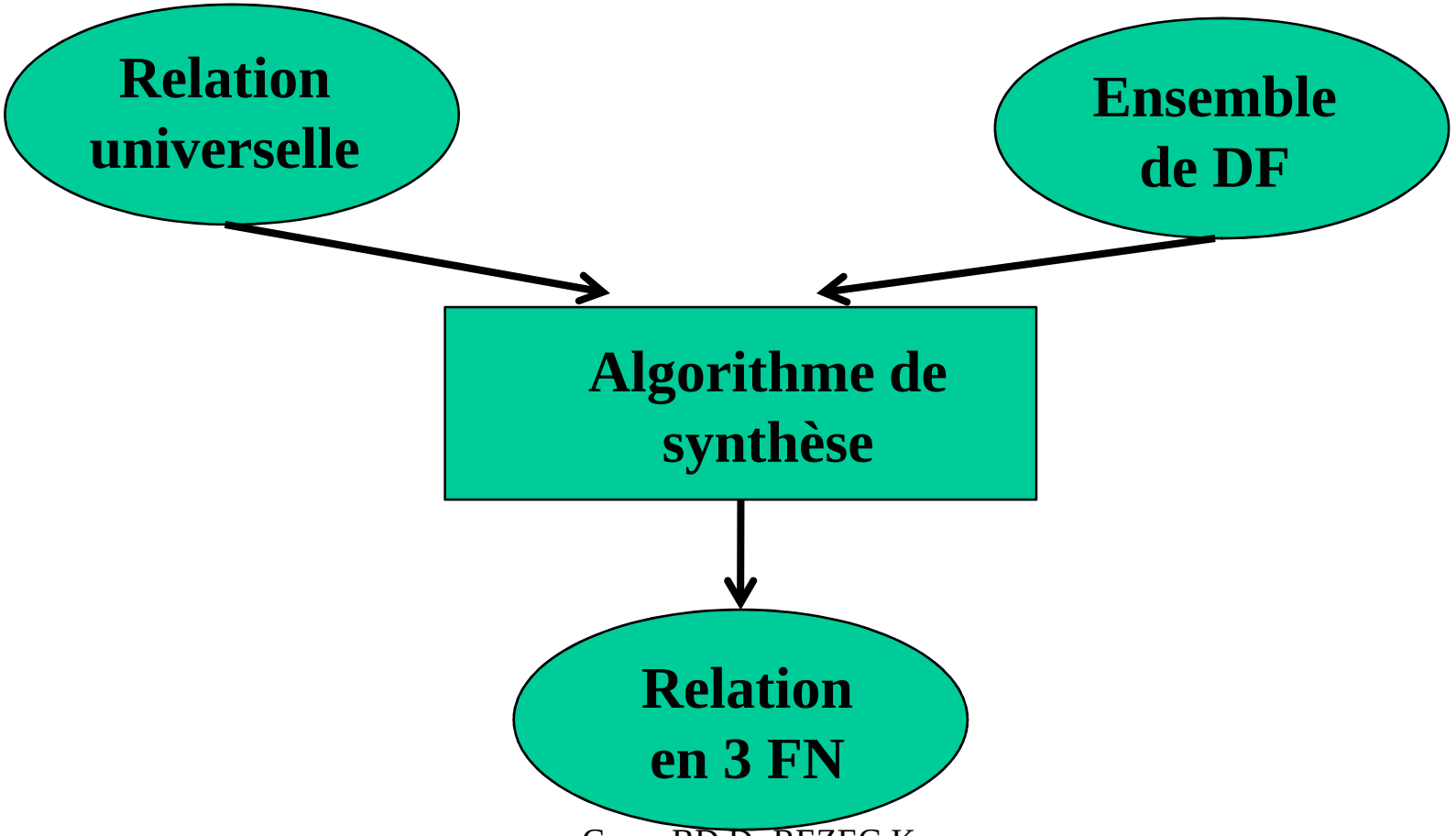
DF2: nom, produit → prix

Alors la clé de la relation Fournisseur est [nom, produit].

Cette relation n'est pas en FNBC, car dans la DF1, la partie gauche "nom" n'est pas une clé entière.

Algorithme de synthèse (1)

- En appliquant les étapes de cet algorithme, vous pourrez déduire un schéma de relation en 3 FN.



Algorithme de synthèse (2)

- Une relation universelle : est une relation qui contient tous les attributs du domaine d'étude.

- Couverture minimale (irréductible): est un sous ensemble minimum de DF élémentaires permettant de générer toutes les autres.
  - Théorème: Tout ensemble de DF admet une couverture minimale, en général non unique

Algorithme de synthèse (3)

- Exemple :
- 1- code_mod → cod_filière
- 2- cod_filière → libélé_filière
- 3- ~~Code_mod → libélé_filière~~ **Transitivité de 1 et 2**
- 4- Jour, heure, local → num_ens
- 5- ~~Jour, heure, local → cod_filière~~ **Transitivité de 1 et 9**
- 6- Jour, heure, local → section
- 7- Jour, heure, local → groupe
- 8- Jour, heure, local → an_étude
- 9- Jour, heure, local → cod_mod
- 10 - ~~Jour, heure, local , num_ens → cod-filière~~

Augmentation de 5

Algorithme de synthèse (4)

- La couverture minimale est la suivante :
- 1- code_mod → cod_filière
- 2- cod_filière → libélé_filière
- 3- Jour, heure, local → num_ens
- 4- Jour, heure, local → section
- 5- Jour, heure, local → groupe
- 6- Jour, heure, local → an_étude
- 7- Jour, heure, local → cod_mod

Algorithme de synthèse (5)

- Etapes de l'algorithme de synthèse
  - Trouver l'ensemble IRR(F) contenant la couverture irréductible (minimale) de F;
  - Partitionner l'ensemble IRR(F) en sous ensemble de Fi, tels que toutes les DFs de Fi ont la même partie gauche;
  - Pour chaque ensemble Fi de DFs, construire une relation composée de tous les attributs formant les DFs de Fi, la clé de la relation sera la partie gauche commune à ces DFs;
  - Traiter les attributs isolés

Algorithme de synthèse (7)

- Exercice
  - 1- Code_mod → libellé_mod
  - 2- Code_mod → an_etude
  - 3- Code_mod → libellé_filière
  - 4- code_filière → libellé_filière
  - 5- matricule → nom_et
  - 6- matricule → adresse-et
  - 7- matricule → dat_nai_et
  - 8- num_ens → nom_ens
  - 9- num_ens → dat_nai_ens
  - 10- num_ens → grade
  - 11-num_ens→situation_familiale
  - 12- num_ens → salaire
  - 13- grade → nbre_heures
  - 14- grade → salaire
  - 15- cod_mod → coef
  - 16- cod_mod → cod_filière
  - 17- matricule → prenom_et
  - 18- num_ens → prenom_ens
  - 19- num_ens → adresse_ens
  - 20- num_ens, grade→nbre_heures
  - 21- num_ens, grade → salaire
  - 22- num_ens → nbre_heures
  - 23- matricule → section
  - 24- matricule → groupe
  - 25-cod_mod,matricule→cod_filière

Algorithme de synthèse (8)

- Concevoir un schéma relationnel en 3FN, à partir de cet ensemble de DFs en appliquant l'algorithme de synthèse.

Chapitre 4

Manipulation de bases de données relationnelles

Algèbre relationnelle

- Les opérandes sont les relations ou variables (qui représentent des relations)
- Les opérateurs sont conçus afin de répondre aux besoins des utilisateurs
- Le résultat est une algèbre qui peut être utilisé comme langage de requêtes

Opérateurs de base

- Union, intersection, et la différence.
  - Des opérations ensemblistes qui imposent que les opérandes aient le même schéma.
- Sélection: garder certains n-uplets.
- Projection: garder certaines colonnes.
- Produits et jointure.
- Re-nomage des relations et attributs.

L'algèbre relationnelle
Opérateurs unaires

- Affectation : opération qui consiste à transférer des tuples d'une table dans une autre table. Cette opération est notée ←

- Restriction : Opération qui consiste à supprimer les tuples d'une relation ne satisfaisant pas la condition précisée. Cette opération est notée s

- Projection : Opération qui consiste à supprimer des attributs d'une relation et à éliminer les tuples en double apparaissant dans la nouvelle relation. Cette opération est notée π .

L'algèbre relationnelle
L'affectation

L'affectation permet de sauvegarder le résultat d'une expression de recherche ou bien de renommer une relation et ses attributs.

<u>Notation</u> : ←

<u>Représentation graphique</u> : ↑

<u>Exemple</u> :

R	A	B	C
	a b c	d e f	1 2 3

S(D,E,F) ← R donne :

S	D	E	F
	a b c	d e f	1 2 3

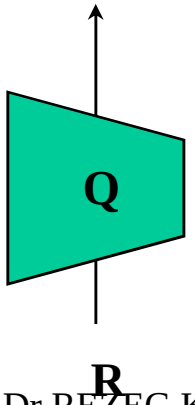
L'algèbre relationnelle
La restriction (1)

La restriction (ou sélection) d'une relation R par une qualification Q est une relation R' de même schéma dont les tuples sont ceux de R qui satisfont à la qualification.

Q peut s'exprimer à l'aide de constante, d'attributs, de comparateurs (>, >=, <, <=,=, <>) et d'opérateurs logiques (, ,)

<u>Notation</u> : σ_Q (R) ou select $_Q$ (R)

<u>Représentation graphique</u> :



L’algèbre relationnelle
La restriction (2)

R	A	B	C
	a b c	d e f	1 2 3

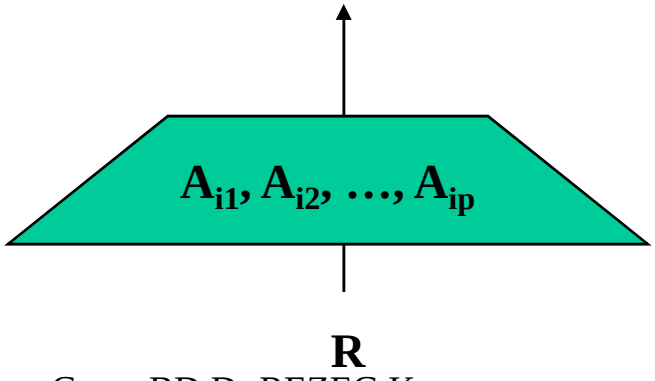
Select (R) C < 4 et A <> ‘a’	A	B	C
	b c	e f	2 3

L'algèbre relationnelle
La projection (1)

La projection d'une relation R de schéma R(A1, A2, …, An) sur les attributs $A_{i1}, A_{i2}, …, A_{ip}$ avec avec ij != ik et p<= n est une relation R'($A_{i1}, A_{i2}, …, A_{ip}$) dont Les tuples sont obtenus par élimination des attributs de R n'appartenant pas à R' et par suppression des doublons

<u>Notation</u> : $\pi_{Ai1, Ai2, …, Aip}$ (R) ou proj $_{Ai1, Ai2, …, Aip}$ (R)

<u>Représentation graphique</u> :



L'algèbre relationnelle
La projection (2)

Exemple :

R	A	B	C
	a b c	d e f	1 2 3

$proj_{A,B}(R)$	A	B
	a b c	d e f

$proj_{A}(R)$	A
	a b c

L'algèbre relationnelle
Opérateurs binaires ayant le même schéma

- Union : opération portant sur deux relations ayant le même schéma et construisant une troisième relation constituée des tuples appartenant à chaque relation. Les tuples en double sont éliminer

- Intersection : Opération portant sur deux relations ayant le même schéma et construisant une troisième relation dont les tuples sont constitués de ceux appartenant au deux relations

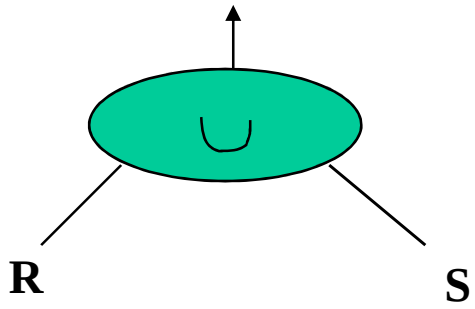
- Différence relationnelle : Opération portant sur deux relations ayant le même schéma et construisant une troisième relation dont les tuples sont constitués de ceux ne se trouvant que dans une seule relation

L'algèbre relationnelle
L'union (1)

L'union de deux relations R et S de même schéma est une relation T de même schéma contenant l'ensemble des tuples appartenant à R, à S ou aux deux.

<u>Notation</u> : R ∪ S ou union (R, S)

<u>Représentation graphique</u> :



L’algèbre relationnelle
L’union (2)

R	A	B	C
	a b c	d e f	1 2 3

T	A	B	C
	a s c	d e d	1 4 3

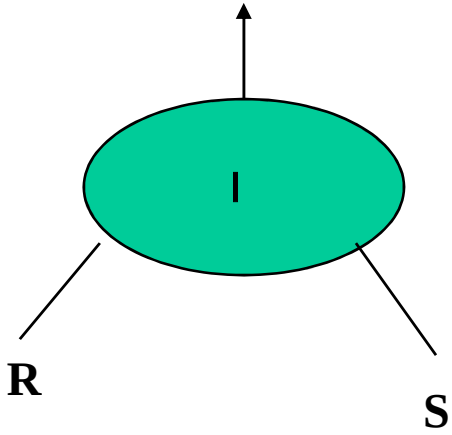
Union (R,T)	A	B	C
	a b c s c	d e f e d	1 2 3 4 3

L'algèbre relationnelle
L'intersection(1)

L'intersection de deux relations R et S de même schéma est une relation T de même schéma contenant l'ensemble des tuples appartenant à la fois à R et à S.

<u>Notation</u> : R I S ou inter (R,S)

<u>Représentation graphique</u> :



L’algèbre relationnelle
L’intersection(2)

R	A	B	C
	a b c	d e f	1 2 3

T	A	B	C
	a s c	d e d	1 4 3

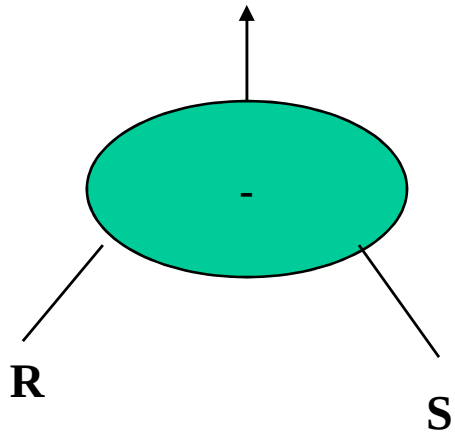
Inter (R,T)	A	B	C
	a	d	1

L'algèbre relationnelle
La différence (1)

La différence entre deux relations R et S de même schéma est une relation T de même schéma contenant l'ensemble des tuples appartenant à R et n'appartenant pas à S.

<u>Notation</u> : R – S ou minus (R, S)

<u>Représentation graphique</u> :



L’algèbre relationnelle
La différence (2)

R	A	B	C
	a b c	d e f	1 2 3

T	A	B	C
	a s c	d e d	1 4 3

R- T	A	B	C
	b c	e f	2 3

L'algèbre relationnelle
Opérateurs binaires ayant un schéma différent

- Le produit cartésien : opération sur deux relations de schéma différents construisant une troisième relation constituée des attributs appartenant à chaque relation et dont les tuples sont constitués de toutes les concaténations des tuples des deux relations.

- La jointure : opération qui consiste à faire le produit cartésien de deux relations, puis à supprimer les tuples ne satisfaisant pas une condition portant sur un attribut de la première relation et sur un attribut de la seconde.

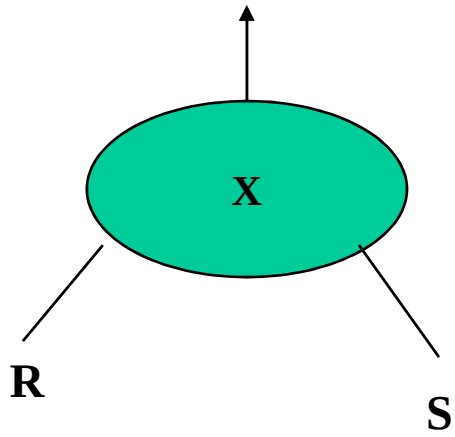
- La division : opération sur deux relations de schéma différents construisant une troisième relations constituée de tous les tuples qui concaténé à chaque tuple de la deuxième relation, donnent toujours un tuple de la première relation.

L'algèbre relationnelle
Le produit cartésien (1)

Le produit cartésien de deux relations R et S de schémas quelconques est une relation T ayant pour attributs la concaténation des attributs de R et de S et dont les tuples Sont constitués de toutes les concaténations d'un tuple de R à un tuple de S.

<u>Notation</u> : R x S ou product(R,S)

<u>Représentation graphique</u> :



L'algèbre relationnelle
Le produit cartésien (2)

R	A	B	C
	a b c	d e f	1 2 3

S	D	E
	a b	d e

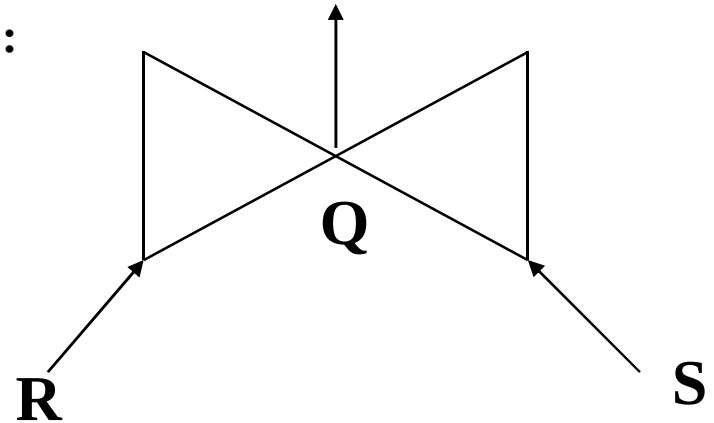
R x S	A	B	C	D	E
	a a b b c c	d d e e f f	1 1 2 2 3 3	a b a b a b	d e d e d e

L'algèbre relationnelle
La θ jointure (1)

La θ jointure de deux relations R et S selon une qualification Q est l'ensemble des Tuples du produit cartésien R qui satisfont à la qualification Q. Il s'agit donc de la Restriction selon Q de R x S, c'est à dire σ_Q (R X S)

<u>Notation</u> : R $\bowtie_Q$ **S ou joint $_Q$ (R,S)**

<u>Représentation graphique</u> :



L'algèbre relationnelle
La θ jointure (2)

R	A	B	C
	a b c	d e f	b g c

S	D	E
	f b	d e

R ⋈ B<D et A != C S	A	B	C	D	E
	a b	d e	b g	f f	d d

L'algèbre relationnelle
L'équi-jointure

L'équi-jointure de deux relations R et S est une ⍴ jointure avec pour qualification Q l'égalité entre deux colonnes, c'est-à-dire R $\bowtie_{Ai = Bj}$ S avec Ai et Bj, deux attributs de R et de S respectivement

R	A	B	C
	a	d	d
	b	e	g
	c	f	c

S	D	E
	d	f
	b	e

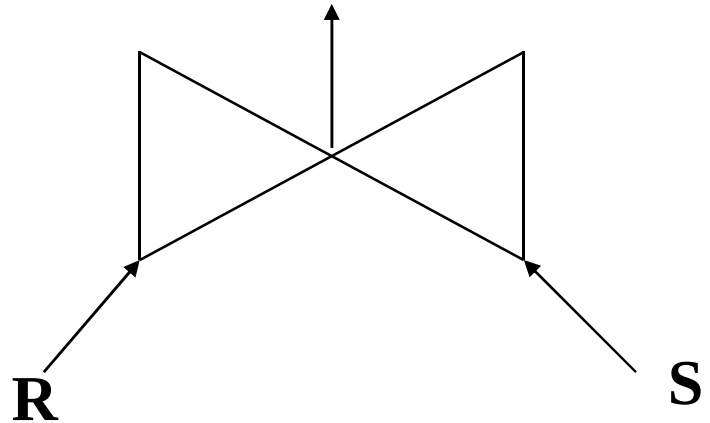
R $\bowtie_{B= D}$ S	A	B	C	D	E
	a	d	d	d	f

L'algèbre relationnelle
La jointure naturelle (1)

La jointure naturelle de deux relations R et S est une équi-jointure sur tous les attributs de même nom dans R et dans S, suivie de la projection qui permet de ne conserver qu'un seul des cas attributs égaux de même nom.

<u>Notation</u> : R ⋈ **S ou joint (R,S)**

<u>Représentation graphique</u> :



L’algèbre relationnelle
La jointure naturelle (2)

R	A	B	C
	a	d	s
	b	e	g
	c	f	c

S	A	B	D
	a	d	d
	a	d	g
	c	f	c

R ⋈ S	A	B	C	D
	a	d	s	d
	a	d	s	g
	c	f	c	c

Problèmes liés à la jointure (1)

La jointure n'inclut les tuples que s'il y a égalité entre deux colonnes

Exemple : on désire la liste de tous les départements de l'entreprise avec les employés Associés.

Dpt	Did	Dnom	Dville
	D1	Achats	Amiens
	D2	Recherche	Boves
	D3	Ventes	Ailly
	D4	Informatique	Dreuil

Emp	Eid	Enom	Epren	Did
	E10	Black	John	D3
	E20	White	Bob	D3
	E30	Léger	Ferdinand	D2
	E40	Flam	Captain	D3
	E50	Albator	Roger	D1
	E60	DreamBox	René	D1

Problèmes liés à la jointure (2)

Dpt ⋈ Emp	Did	Dnom	Dville	Eid	Enom	Epren
	D1	Achats	Amiens	E50	Albator	Roger
	D1	Achats	Amiens	E60	Dreambox	René
	D2	Recherche	Boves	E30	Léger	Ferdinand
	D3	Ventes	Ailly	E10	Black	John
	D3	Ventes	Ailly	E40	Flam	Captain
	D3	Ventes	Ailly	E20	White	Bob

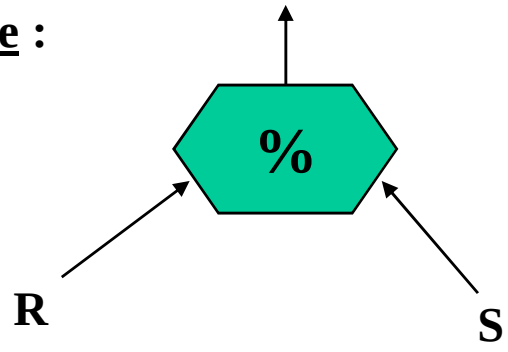
Remarque : il nous manque le département informatique !!!

L'algèbre relationnelle
La division (1)

La division (ou quotient) de la relation R (A1, A2, …, An) par la (sous-)relation S de schéma S(A p+1, …, An) est la relation de schéma T (A1, A2, …, Ap) formée de Tous les tuples qui, concaténés à chaque tuple de S, donnent toujours un tuple de R.

<u>Notation</u> : R / S ou div(R,S)

<u>Représentation graphique</u> :



L’algèbre relationnelle
La division (2)

R	A	B	C	D
	a	b	c	d
	a	b	e	f
	b	c	e	f
	e	d	c	d
	e	d	e	f
	a	b	d	e

S	C	D
	c	d
	e	f

R / S	A	B
	a	b
	e	d

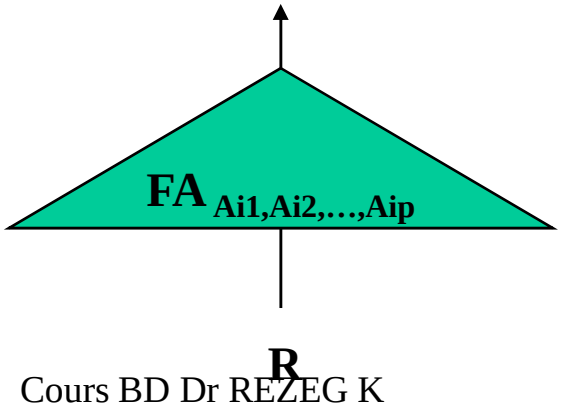
Agrégats

Un agrégat est un partitionnement horizontal d ’une relation selon des valeurs d ’attributs, suivi d ’un regroupement par une fonction de calcul

<u>Notation</u> : Fonction_d ’Agrégat $_{Ai1,Ai2,\ldots,Aip}$ (R)

Par exemple : Compte $_{A,B,C}$ (R)

<u>Représentation graphique</u> :



Agrégats : Exemple

R	A	B	C
	a	b	10
	d	a	15
	c	b	5
	b	g	8

Compte (R)	Compte
	4

Moyenne (R , C)	Moyenne
	9,5

Compte $_B$ (R, C)	B	Compte
	b	2
	a	1
	g	1

Somme $_B$ (R, C)	B	Somme
	b	15
	a	15
	g	8

Expressions de l'algèbre relationnelle(1)

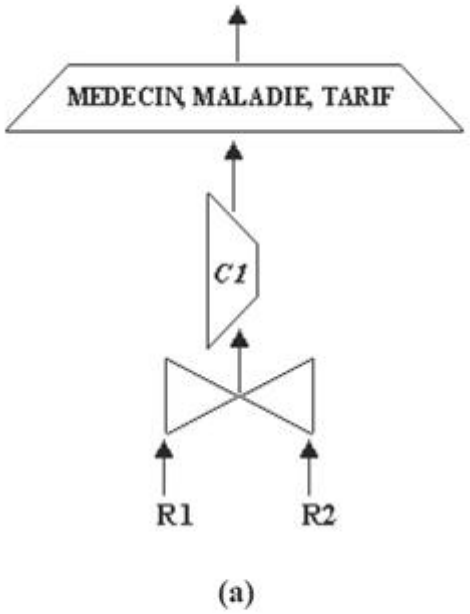
Les opérations algébriques peuvent être combinées pour former des expressions de l'algèbre relationnelle. Il sera donc possible de composer la plupart des questions que l'on peut poser à une base de données relationnelle. Ainsi, ces questions peuvent être exprimées à l'aide de successions des opérations UNION, DIFFERENCE, JOINTURE, SELECTION, PROJECTION. La représentation graphique de ces opérations permet de composer des arbres d'opérations relationnelles.

Expressions de l'algèbre relationnelle(2)

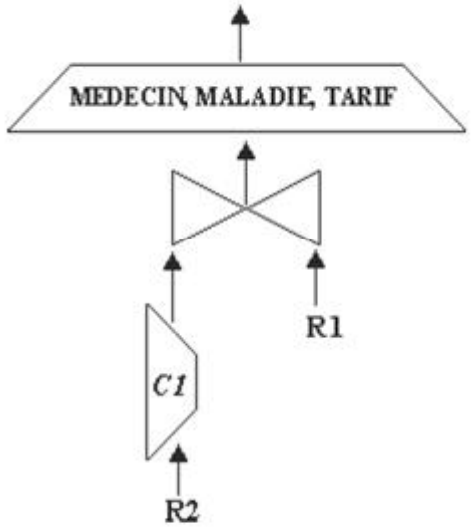
Exemple : soit la base de données composée des relations suivantes :

- R1(MEDECIN , MALADIE , TARIF)
- R2(NUMERO , MALADE , MALADIE)
- La relation R1 représente les maladies qui peuvent être examinées par un médecin et le tarif de la consultation chez ce médecin pour chaque maladie.
- La relation R2 représente les maladies pour lesquelles un malade souhaite être examiné.
- La réponse à la question suivante : "Quels sont les noms des médecins pouvant examiner le malade "Adel" et les prix de leurs consultations ainsi que les maladies à examiner " peut être exprimée à l'aide de l'un des deux arbres suivants :

Expressions de l'algèbre relationnelle(3)



(a)



(b)

Expressions de l’algèbre relationnelle(4)

- La condition C1 est : Malade = ‘‘Adel’’
- Un arbre d’opérations s’interprète de bas en haut. Les expressions algébriques correspondant chacun des deux arbres précédents sont respectivement :
- a) PROJECT médecin, maladie, tarif (SELECT malade = ’’Adel’’ (JOIN (R1, R2)))
- b) PROJECT médecin, maladie, tarif (JOIN (SELECT malade = ’’Adel’’ (R2), R1))
- (b) est plus efficace que (a) puisque on applique d’abord le SELECT sur R2 qui donnera une relation intermédiaire contenant uniquement les tuples relatifs au malade "Adel" (c.a.d. 2) et qu’on utilisera pour calculer le JOIN avec R1.

Chapitre 4

Le langage SQL
(Structured Query Language)

Introduction

- SQL est le langage informatique standard pour la communication avec les SGBD relationnelles. Il peut être logiquement vu comme étant composé de 3 sous langages: un Langage de Manipulation de Données (LMD), un Langage de Description de Données (LDD) et un Langage de Contrôle de Données (LCD).

- Interrogation en SQL

L'interrogation de relation s'exprime à l'aide de la commande Select, à ne pas confondre avec la sélection algébrique.

Projection

SELECT liste d'attributs

FROM nom de relation

- Exemple : soit la relation Produit (code, desig, PU), et soit l'expression en algèbre relationnelle suivante: Project $_{Desig, PU}$ (produit), elle se traduit en SQL comme suit: SELECT desig, PU FROM Produit

- RQ: SQL n'élimine pas les tuples en double à moins que ceci soit explicitement demandé par un mot clé du langage qui est : DISTINCT (ou parfois UNIQUE) → SELECT UNIQUE

Sélection

SELECT *

FROM nom de relation

WHERE condition ;

- " * " dénote toutes les colonnes de la relation.
- Exemples:
- 1) SELECT $_{(PU<=250)}$ (produit) → SELECT * FROM Produit WHERE (PU<= 250)
- 2) "Quel sont les produits dont le nom commence par 'P' et dont le prix unitaire est inférieur ou égale à 300?"
- SELECT * FROM Produit
- WHERE (Desig Like "P%") and (PU<= 300)

Composition de la sélection et de la projection

SELECT liste d'attributs

FROM nom de relation

WHERE condition

- Exemple: "liste contenant le code et la désignation des produits dont leur prix unitaire est supérieur ou égal 500"
  → SELECT code, desig

    FROM Produit

    WHERE (PU>= 500)

Produit cartésien

- Le produit cartésien entre deux relations s'exprime comme une jointure sans condition (qualification).
- SELECT * FROM liste de noms de relations

- Exemple: le produit cartésien des relations EMPLOYES et DEPARTEMENTS s'obtient à l'aide de la requête suivante : SELECT * FROM EMPLOYES , DEPARTEMENTS

Jointure

- SELECT * FROM liste de noms de relations WHERE expression de jointure
- Exemples: "Quels sont les numéros et les noms des employés qui travaillent à ‘BATNA’ ?
- SELECT Num_Emp , Nom_Emp
- FROM EMPLOYES , DEPARTEMENTS
- WHERE (EMPLOYES•Num_Dept=DEPARTEMENTS•Num_Dept) AND (Ville = ’BATNA’)

Expression de l'union, de l'intersection et de la différence

- SQL offre des opérateurs ensemblistes qui permettent d'exprimer l'union (UNION), l'intersection (INTERSECT) et la différence (MINUS) à l'aide de requêtes:

< Clause Select > < opérateur ensembliste> <Clause Select>

- Il faut veiller à ce que les opérandes gauche et droit de l'opérateur soit des relations de même schéma ou de schémas compatibles.

La description des données avec SQL (1)

- Création de schéma de relations

La création de schéma de relation consiste à nommer la relation et à en définir les attributs;

- CREATE TABLE nom de relation (nom_attribut1 type_attribut1 [Not NULL| NULL] [nom_attribut2 type_attribut2 [Not NULL| NULL]...)
- type_attribut peut être par exemple: Int: entier signé représenté sur 4 octets; Float: virgule flottante;
- Char (longueur): chaîne de longueur fixe.
- NOT NULL: spécifie qu'une colonne ne doit pas contenir de valeurs NULL (ou indéfinies, inconnues).
- Exemple: CREATE TABLE ETUDIANTS (NUMERO int NOT NULL, NOM CHAR(10), MOYENNE float)

La description des données avec SQL (2)

- Suppression d'une table : DROP TABLE
- DROP TABLE <nom de table> ; supprime le contenu de la relation ainsi que sa définition, c. a d. son schéma.

- Modification de schémas de relations:
- La commande ALTER TABLE permet de modifier le schéma d'une relation en y ajoutant de attributs ou des contraintes, en modifiant la définition d'un attribut, ...la variété des modifications dépend des SGBD. Il faut cependant avoir à l'esprit que la modification de schéma de relations pendant la durée de vie d'une base peut avoir des incidences sur les données et sur les programmes existants.

Instruction SQL de mise à jour

- INSERT : ajout de tuples

L'opération d'insertion permet d'ajouter un ou plusieurs tuples à une relation

- INSERT INTO <nom de relation> [(liste de colonnes)] {VALUES (expression_constante[, expression_constante[, ...]]) | clause select}
- UPDATE: modification de tuples
- UPDATE modifie des valeurs de colonnes dans une ou plusieurs lignes
- UPDATE nom de relation SET colonne 1= {expression| NULL| (clause Select)} [, colonne 2= ...] [FROM ...] [WHERE qualification]

Instruction SQL de mise à jour

- Delete: Suppression de tuples

La suppression peut concerner tous les tuples d'1 relation ou un sous-ensemble de tuple qui vérifie une condition de selection; il fait noter que delete opère sur le contenu d'une relation, c'est-à-dire que le schéma d'une relation persiste après suppression de tous les tuples de la relation.

- DELETE [FROM] nomde relation [,nom de relation[, ...]] [WHERE qualification]

Les fonctions de groupes

Aperçu général

- SQL offre aussi un certain nombre de fonctions dites fonctions de groupes qui peuvent être utilisées dans l'expression d'une requête. Les plus importantes sont :
- AVG : permet de calculer une MOYENNE
- SUM : permet de calculer une SOMME
- COUNT : permet de compter des tuples
- MAX : permet de calculer un maximum
- MIN : permet de calculer un minimum
- GROUP BY : permet de créer des sous-ensembles de tuples
- HAVING : permet de tester si une condition est vérifiée par un groupe de tuples

Tri

- Tri du résultat d'une requête : la clause ORDER BY
- La clause ORDER BY permet de trier le résultat retourné par une requête. L'ordre peut être ascendant (ASC) ou descendant (DESC) et sera fonction du type de ou des attributs qui seront spécifiés comme arguments de cette clause. Les exemples suivants permettent de donner une idée des possibilités de cette clause.
- Q8: Liste des employés du département 10 par ordre décroissant de leur salaire?
- SELECT Num_Dept , Nom_Emp, Salaire FROM EMPLOYES
- WHERE Num_Dept = 10 ORDER BY Salaire DESC

Exemples de requêtes (1)

- Q1:«Quel est le nom, la fonction et le salaire de(s) l'employé(s) ayant le salaire le plus élevé»
- Q2: « Quel est le nombre de fonctions différentes exercées dans le département 30 ?»
- Q3: « Quelle est la moyenne des salaires par département? »
- Q4: « Quels est pour chaque département le salaire annuel (12 mois) moyen de tous ses employés sauf ceux ayant une fonction de 'DIRECTEUR' ou 'PRESIDENT'?
- Q5: Quels sont les numéros des départements ayant plus de 10 employés ?
- Q6:« Quelles sont les différentes fonctions exercées dans l'ensemble des départements et la moyenne des salaires par fonction ?»

Exemples de requêtes (2)

- Q7: « Quelles sont les différentes fonctions exercées dans l'ensemble des départements dont la moyenne des salaires par fonction est supérieure à10.000 ?

- Q8: Liste des employés triés selon un ordre alphabétique de leur fonction et à l'intérieur de chaque fonction les trier selon un salaire décroissant.