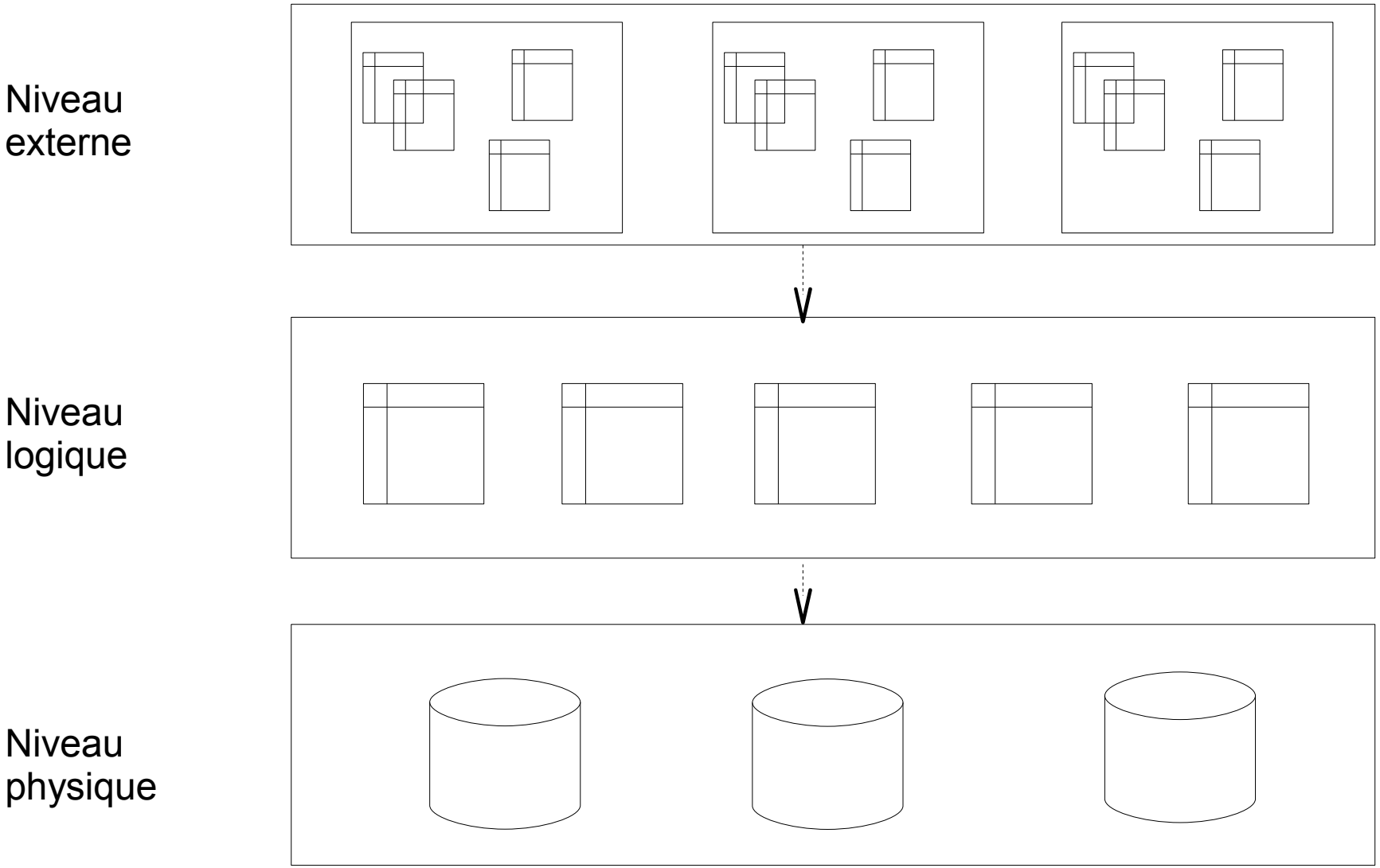
Définitions

- Base de données : ensemble cohérent de données structurées, fiables et partagées entre plusieurs utilisateurs
- Système de gestion de base de données (SGBD / DBMS) : logiciel de haut niveau permettant de gérer, contrôler et manipuler les données

Architecture fonctionnelle d'un SGBD



Niveau physique

- S'appuie sur le système de gestion de fichiers de l'OS pour stocker les données
- Gère le partage des données et les accès concurrents
- Assure la fiabilité des données, notamment suite à des pannes
- La personne responsable de ce niveau est l'administrateur

Niveau logique

- Correspond à la vision des données indépendamment des applications individuelles
- Le langage de définition de données permet de mettre en place les données de manière structurée et correspondant au modèle élaboré
- Le langage de manipulation de données permet la consultation et la mise à jour de ces données

Niveau externe

- Correspond aux vues utilisateurs via leurs applications
- Les accès utilisateurs peuvent être distants, provenir de diverses interfaces et langages
- Chaque utilisateur ne voit que les données qui lui sont utiles
- Il permet le dialogue courant avec la base de données

Repères historiques

- Début des années 60 : les données sont accessibles par les fichiers
- Milieu des années 60 : Bases de données hiérarchiques et réseaux
- 73 : apparition des bases de données relationnelles
- 90 : apparition de XML

BDD : Objectifs

- Assurer l'indépendance physique
- Assurer l'indépendance logique
- Assurer la sécurité et la confidentialité des données
- Garantir la fiabilité des données
- Garantir la cohérence des données
- Pouvoir répondre à des requêtes
- Fournir différents langages d'accès selon l'utilisateur
- Permettre l'administration centralisée

Le modèle relationnel

- Modèle relationnel conçu par E.F. Codd en 1970
- Correspond à une définition rigoureuse des données
- Caractérisé par des règles d'intégrité garantissant la cohérence des données
- Utilise la puissance des opérateurs de l'algèbre relationnelle

Objectifs du modèle relationnel

- Permettre l'independance des programmes par rapport à la représentation interne des données
- Pouvoir traiter les problèmes de cohérence et de redondance des données
- Développer un langage non procédurale de manipulation des données

==> Objectifs atteints : basé sur l'algèbre relationnelle + langage de manipulation SQL

Les concepts

- 3 concepts fondamentaux :
  - Domaine : ensemble de valeurs
  - Relation : tableau à deux dimensions
  - Attribut : nom d'une colonne du tableau
- L'attribut prend ses valeurs dans un domaine
- Deux attributs ne peuvent porter le même nom
- Chaque ligne du tableau est appelée tuple

Schéma relationnel

- Schéma d'une relation : nom de la relation suivi de la liste des attributs et de la définition de leurs domaines
- Schéma relationnel = ensemble des schémas des relations d'une base de données

SALARIE(matricule : entier,
nom : chaîne de caractères,
grade : {employé, agent, cadre, directeur},
salaire : [7000, 240000])

La relation

SALARIE

Attributs →

Matricule	Nom	Grade	Salaire
1025	Dubois	Employé	12000
521	Martin	Directeur	320000
1489	Journade	Agent	10000
653	Lapaj	Cadre	150000
1325	Gayan	Employé	15000

Tuples

- Cardinalité de la relation = nombre de tuples (5)
- Dégré de la relation = nombre d'attributs (4)

EXERCICE

Contraintes d'intégrité

- Expression logique qui doit être toujours vrai dans la base de données
- Permet de contrôler la cohérence des données

- Contrainte de clé
- Contrainte de domaine
- Contrainte de références

Contrainte de clé

- Une clé de relation est un sous-ensemble d'attributs qui permet de caractériser tout enregistrement d'une relation
- Il doit exister un sous-ensemble d’attributs pour lequel deux tuples ne peuvent être identiques
- Les clés candidates sont les attributs (ou groupes) pouvant jouer le rôle de clé. La clé choisie est appelée clé primaire

Contrainte de clé : Exemple

R			
A	B	C	D
a	1	Paris	a
a	3	Toulouse	c
b	5	Lyon	k
c	1	Paris	a
a	2	Paris	c
e	3	Toulouse	b

R(<u>A, B</u>, C, D)

NB : La clé primaire de la relation est soulignée

Clé étrangère

- On appelle clé étrangère tout attribut qui est clé primaire dans une autre relation

Eleve(kEleve, nom, prenom, dateNaissance, kClasse#)

Classe(kClasse, libellé)

NB : Les clés étrangères sont suivies d'un #

EXERCICE

Contraintes d'intégrité (suite)

- Contrainte de domaine : Restriction des valeurs possibles d'un attribut

  Le montant du salaire doit être compris entre 1000 et 3000

- Contrainte de relation : Une clé primaire doit toujours être unique et définie (NOT NULL)

- Contrainte d'intégrité référentielle : Impose que la valeur d'une clé étrangère existe déjà comme valeur de clé primaire

EXERCICE

Théorie de la normalisation

- Permet de définir formellement la qualité des tables au regard du problème posé par la redondance des données
- 2 façons de constituer de bonnes tables :
  - La décomposition revient à décomposer une table contenant l'ensemble des attributs
  - Transformer le modèle entité-relation en formalisme relationnel

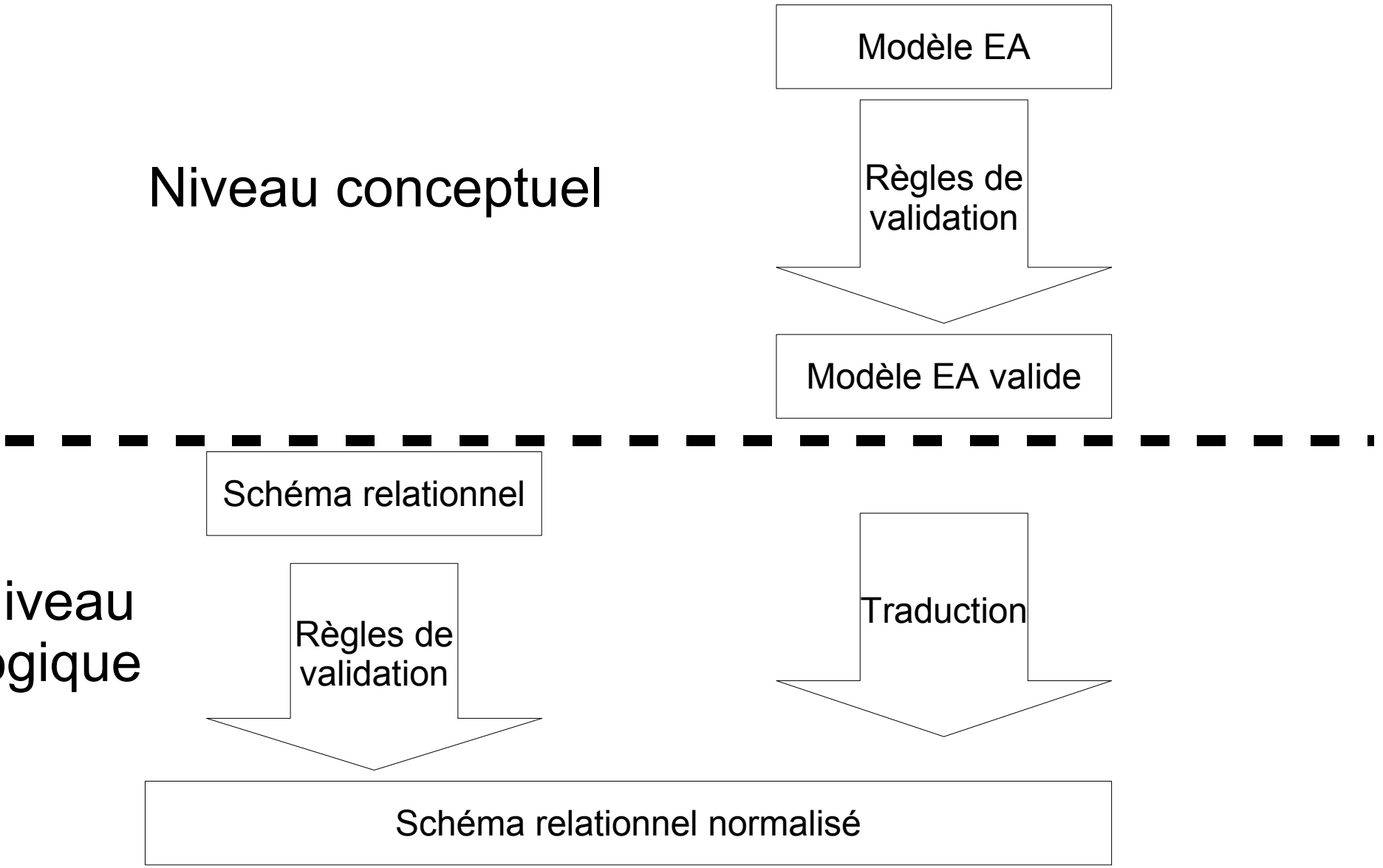
Exemple d'anomalies

stock(numFrs, adrFrs, numProd, puHT, qte)

numFrs	adrFrs	numProd	puHT	qte
3	Toulouse	52	65	10
22	Colomiers	10	15	5
22	Colomiers	25	10	12
3	Toulouse	25	10	5
3	Blagnac	10	15	20

- Changement d'adresse d'un fournisseur
- Insertion d'un produit d'un fournisseur connu qui a changé d'adresse
- Si tous les produits d'un fournisseur sont supprimés, l'adresse du fournisseur est perdue

Normalisation ou traduction EA



Dépendance fonctionnelle

Soit une relation R(X, Y, Z)

Il existe une dépendance fonctionnelle : X --> Y

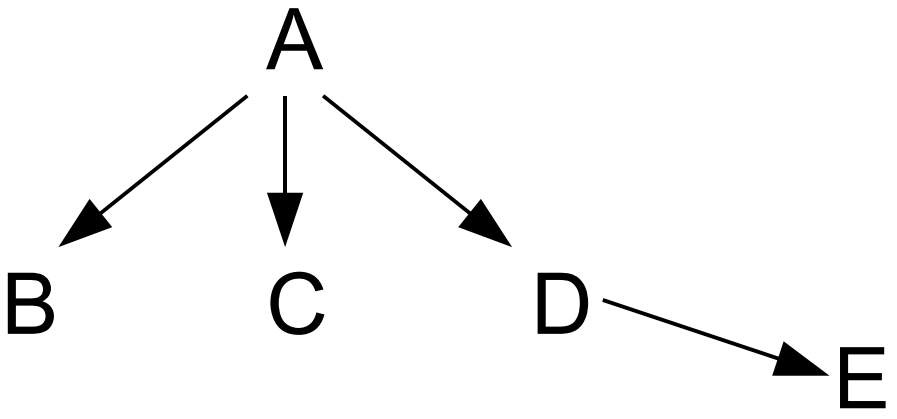
si et seulement si dans R à une même valeur de X correspond toujours une même valeur de Y

X et Y peuvent être des ensembles d'attributs d'une relation

Exemple : numFrs --> nomFrs, adrFrs

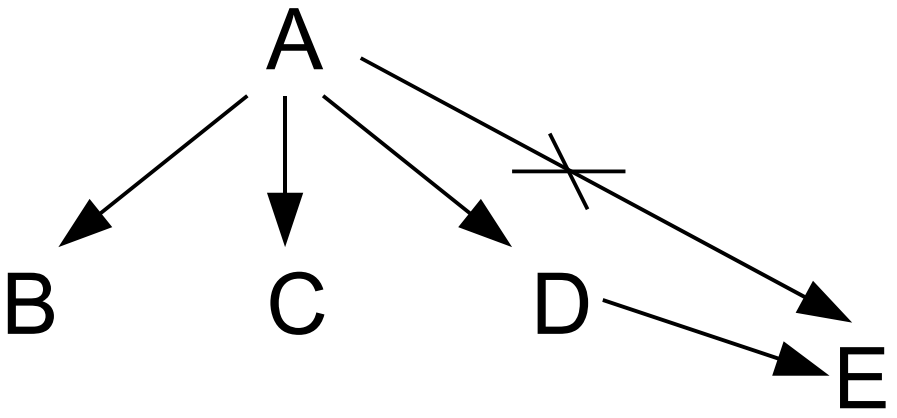
Graphe des DF

- Ce graphe permet de déterminer les clés de la relation



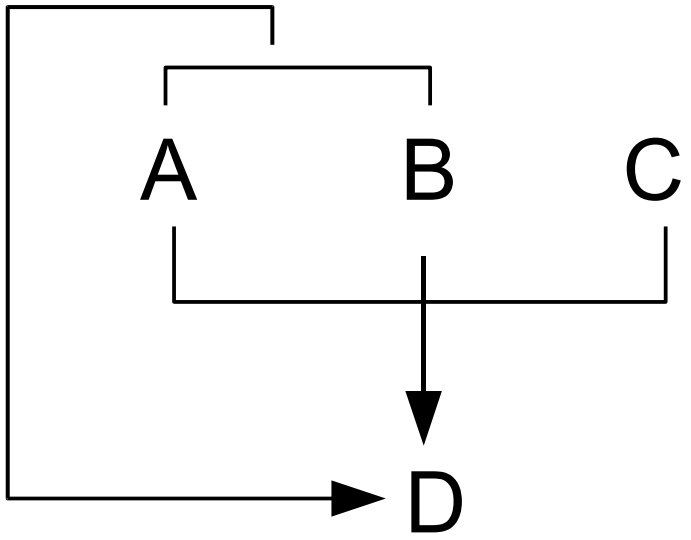
- Le graphe doit être minimum :
  - pas de DF déduite
  - que des DF élémentaires

DF déduites



- A --> E est une DF déduite car il existe un autre chemin A --> D --> E

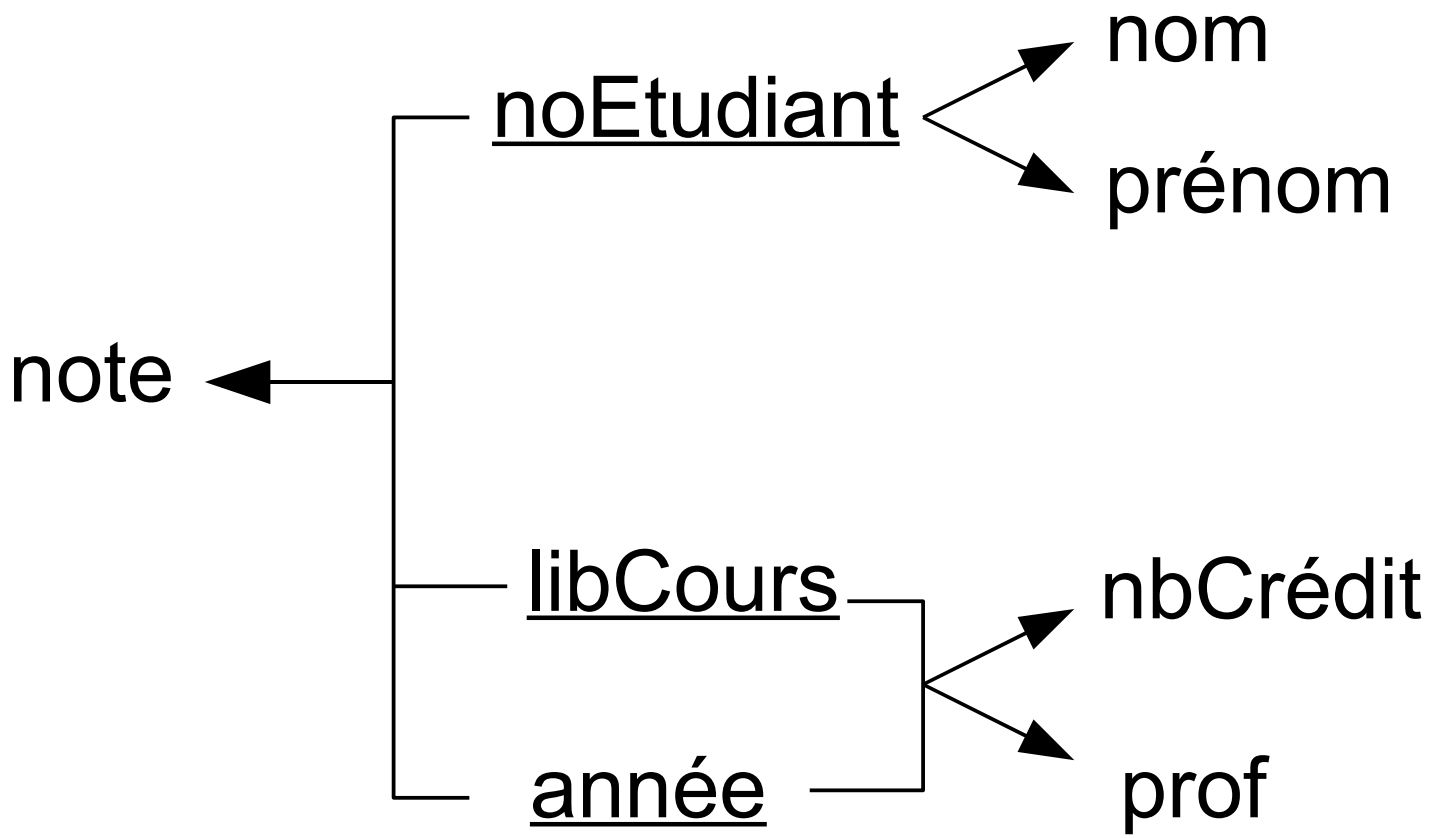
DF non élémentaires



- (A, B, C) --> D est non élémentaire
  car (A, B) --> D existe

Exemple

R(noEtudiant, nom, prénom, libCours, année, nbCrédit, prof, note)



EXERCICE

Normalisation par décomposition

- Si une relation présente des problèmes de redondances il faut la décomposer en plusieurs relations
- La décomposition devra s'effectuer
  - sans perte d'informations
  - sans perte de dépendances fonctionnelles
- Toute DF doit être dans une des relations obtenues

Forme normale

- Une forme normale désigne un type de relation entre deux entités dans une base de données relationnelle.
- La normalisation des modèles de données
  - permet de vérifier la robustesse de leur conception
  - permet d'améliorer la modélisation (obtenir une meilleure représentation)
  - permet de faciliter la mémorisation des données (et donc éviter la redondance et les problèmes sous-jacents de mise à jour ou de cohérence)
- La normalisation s'applique à toutes les entités et aux relations porteuses de propriétés.

Les formes normales

- 1FN : la table ne contient que des attributs atomiques (non décomposables)
- 2FN : il n'existe pas de DF entre une partie de la clé et une colonne non clé
- 3FN : il n'existe aucune DF entre les colonnes non clé

La plupart des tables en 3FN ne comportent plus de redondance

Première forme normale

- Chaque attribut doit contenir une valeur atomique (non composée)
- Tous les attributs doivent être élémentaires et non répétitifs

Exemple :

stock(numFrs, adrFrs, numProd, puHT, qte)

Exemple

Dans la relation STAGIAIRE, on veut conserver 3 stages, on a donc répété 3 fois le code stage.

STAGIAIRE				
CODE-S	NOM-S	CODE-F	CODE-F	CODE-F
89001	ALPHA	CAD21	CAD22	
90001	BRIN	PF12	PF15	SIB123
92003	CACHOU	DOM11		
89007	DUPONT	DOM12		
91010	MARTIN	INF11		

Exemple

Cette relation n'est pas en première forme normale ; le passage en 1NF donne la relation

STAGIAIRE		
CODE-S	NOM-S	CODE-F
89001	ALPHA	CAD21
89001	ALPHA	CAD22
90001	BRIN	PF12
90001	BRIN	PF15
90001	BRIN	SIB123

Le nombre de tuples a été augmenté et on introduit une importante redondance (les noms sont répétés dans chaque nouveau tuple).

EXERCICE

Deuxieme forme normale

- La relation doit être en première forme normale
- Chaque attribut qui n'appartient pas à la clé ne dépend pas uniquement d'une partie de la clé
- Permet d'éliminer les attributs qui ne décrivent pas "l'objet"

Exemple :

Stock(numFrs, villeFrs, numProd, puHT, qte)

Fournisseur(numFrs, villeFrs)
Produit(numProd, puHT, qte)

Exemple

Pour différencier deux tuples de la relation STAGIAIRE mise en 1NF, il faut associer les 3 attributs CODE_S, NOM_S, CODE_F, qui constituent donc la clé primaire. Or, le nom du stagiaire ne dépend que du code stagiaire et le code de la formation est totalement indépendant du stagiaire (il existe indépendamment des stagiaires qui suivent la formation). Cette relation n'est donc pas en deuxième forme normale.

Exemple

- Pour mettre la relation en deuxième forme normale, on est amené à créer deux nouvelles relations :
- FORMATION : qui permettra de connaître les formations
- SUIVRE : qui permettra de retrouver l'intégralité des informations que l'on avait avant la mise en 2NF, c'est à dire le lien entre un stagiaire et la formation auquel il a participé.

Exemple

STAGIAIRE	
CODE-S	NOM-S
89001	ALPHA
90001	BRIN
92003	CACHOU
89007	DUPONT
91010	MARTIN

FORMATION	
CODE-F	LIBELLE
CAD21	Cadastre administratif
CAD22	Cadastre tecnique
PF12	Publicité Foncière publicité
PF15	Publicité Foncière
SIB123	Bureatique
DOM11	Domaine Gestion
DOM12	Domaine evaluation
INF11	Informatique Générale

SUIVRE	
CODE-S	CODE-F
89001	CAD21
89001	CAD22
90001	PF12
90001	PF15
90001	SIB123
92003	DOM11
89007	DOM12
91010	INF11

EXERCICE

Troisième forme normale

- La relation doit être en deuxième forme normale
- Les attributs qui ne font pas partie de la clé ne dépendent pas d'attributs ne faisant pas non plus partie de la clé
- Permet déliminer des sous-relations incluses

Exemple :

Fournisseur(numFrs, ville, pays)

Fournisseur(numFrs, ville)
Lieu(ville, pays)

Exemple

- Considérons la relation PRODUIT qui a pour schéma :
- PRODUIT (CODE-PROD, LIBELLE, PRIX,CODE-TVA, TAUX-TVA)
- L'attribut TAUX_TVA dépend de l'attribut CODE_TVA qui n'est pas un élément de la clé ; cette relation n'est donc pas en troisième forme normale !
- Le passage en 3NF conduit à créer une relation nouvelle, TVA de schéma :
- TVA : (CODE_TVA, TAUX_TVA)

EXERCICE

Forme normale de Boyce-Codd

- Si une relation en troisième forme normale a une clé concaténée, aucun des attributs de cette clé ne doit être en dépendance fonctionnelle d'un autre attribut
- Cette normalisation conduit parfois à décomposer une relation en deux relations plus simples.

Exemple

Considérons la relation Vins(cru, pays, région) avec les dépendances fonctionnelles supposées :

région --> pays

(cru, pays) --> région

Cru	Pays	Région
Chenas	France	Beaujolais
Pomerol	France	Bordeaux
Chablis	France	Bourgogne
Brouilly	France	Beaujolais
Chablis	Etats-Unis	Californie

Exemple

- Cette relation est bien en troisième forme normale car aucun attribut non clé ne dépend d'une partie de la clé ou d'un attribut non clé. Cependant, on y trouve de nombreuses redondances
- La relation Vins pourra être décomposée en deux relations :
  - Crus (cru, région#)
  - Régions (région, pays)
- La dépendance fonctionnelle (cru, pays)-->région est perdue mais elle peut être recomposée par jointure.

Résumé des formes normales

1NF → 2NF → 3NF → BCNF

Éliminer toute dépendance non totale des attributs sur la clé primaire

Éliminer toute dépendance entre deux attributs ne faisant pas partie de la clé primaire

Éliminer toute dépendance à partir d'un attribut ordinaire vers une partie de clé primaire

Quatrième forme normale

- Pour toute relation de dimension n en forme normale de Boyce-Codd, les relations de dimension n-1 construites sur sa collection doivent avoir un sens. Il ne doit pas être possible de reconstituer les occurrences de la relation de dimension n par jointure de deux relations de dimension n-1
- Cette normalisation conduit parfois à décomposer une relation complexe en deux relations plus simples.

Exemple

Employe(noEmploye, stage, sport)

noEmploye	stage	sport
10	informatique	tennis
10	informatique	biathlon
10	cuisine	tennis
10	cuisine	biathlon

R1(noEmploye, stage)

R2(noEmploye, sport)