Plan du cours

1. Introduction
2. Modèle conceptuel de données Entité-Association
3. **Modèle relationnel de données**
    - **Concepts du modèle relationnel**
    - Redondance des données et normalisation
    - Passage du modèle entité-association au relationnel
    - Langages de manipulation des données (LMD)
4. Le langage SQL
5. Le langage PL/SQL
6. Transactions et concurrence d'accès

Modèle relationnel : un peu d'histoire

E.F. Codd dans *CACM 1970*

« A Relational Model of Data for Large Shared Databanks »

Le modèle relationnel est un **modèle au niveau logique** associé aux SGBD relationnels (cf. processus de conception de BD)

Base de données vue par l'utilisateur =

Ensemble de **tableaux** (ou de **tables**)

Nombreux travaux fondamentaux sur les :

- méthodes de conception de BDR
- langages de manipulation des données (algèbre et calcul relationnel)

Modèle relationnel : un peu d'histoire

SGBD relationnels

1976 : Premières réalisations (SYSTEM-R, INGRES)

1980 : Premières commercialisations

2003 : marché inondé (Oracle, Sybase, Informix, MS Access…)

Modèle relationnel : deux parties « théoriques »

- Concepts du modèle (*table, attribut, domaine* …)
- Langage de manipulation des données
    - langage algébrique ou algèbre relationnelle
    - langage prédicatif (formules du CP1) : CRT

Introduction intuitive

Une Base de Données Relationnelle (BDR) peut être vue comme un ensemble de **tables** ou de **relations**.

Une table est un ensemble de lignes (**tuples**) ou de colonnes (**attributs**)

Exemple de BDR composée de 3 tables:

- **fournisseur** est une table contenant le numéro (nof), le nom (nomf) et la ville (ville) de chaque fournisseur
- **pièce** est une table contenant le numéro (nop), le nom (nomp) et le prix (prix) de chaque pièce
- **vente** est une table indiquant qu'une pièce (nop) est vendue par un fournisseur (nof)

Table fournisseur

nof	nomf	ville
1	Girard	Lyon
2	Blanc	Paris
3	Merlin	Nancy

Table vente

nop	nof
1	1
1	2
2	2
2	3
3	1
3	2
3	3

Table pièce

nop	nomp	prix
1	vis	1.5
2	écrou	2
3	boulon	2.5

Une base de données relationnelle constituée de 3 tables (relations)

Formalisation du modèle relationnel de données (*théorie des ensembles)*

- Un **domaine** est un ensemble de valeurs
  - ex : L'ensemble des nombres entiers (Z) ; l'ensemble des chaînes de caractères de longueur 50 ; {jaune, vert, bleu} ; {x,y,z}

- Le **produit cartésien** des domaines $D_1, D_2, \ldots, D_n$ noté $D_1 \times D_2 \times \ldots \times D_n$ est l'ensemble des tuples $(v_1, v_2, \ldots, v_n)$ tel que $v_i \in D_i$, $1 \leq i \leq n$

ex : $n=2$, $D_1 = \{1,2\}$, $D_2 = \{x,y\}$, $D_1 \times D_2 = \{(1,x), (1,y), (2,x), (2,y)\}$

➢ Une **relation** (ou **table)** est un sous-ensemble du produit cartésien d'un ou plusieurs domaines : $R \subseteq D_1 \times D_2 \times \ldots \times D_n$

ex : $R = \{(1,x), (1,y), (2,y)\}$

Schéma de relation

Le **schéma** d'une relation $R \subseteq D_1 x D_2 x \ldots x D_n$ comprend :

- son **nom** R
- le nom de ses **attributs** A_1, A_2, … A_n correspondant aux composantes d'un tuple
- Di : **domaine** de l'attribut A_i c'est à dire l'ensemble des valeurs possibles de A_i
- Un ensemble de contraintes d'intégrité (CI)

Schéma abrégé (sans préciser les domaines ni les CI) : **R (A_1 , A_2, … A_n)**

Deux schémas de relations $R(A_1, A_2 \ldots A_k)$ et $S(B_1, B_2 \ldots B_k)$ sont **compatibles** si A_i et B_i ont le même domaine pour tout i

-> ce sont les **mêmes schémas à des renommages d'attributs près**.

Exemples de schémas de relations

Deux exemples de schémas de relations

- n = 2, D_1=D2={1,2…,6}, relation **est-le-triple-de**
  - ✓ attribut nombre pour la première composante
  - ✓ attribut triple pour la seconde composante
  - ✓ schéma : **est-le-triple-de (nombre,triple)**

    domaine(nombre)= D_1 et domaine(triple)= D_2
  - ✓ seuls les "faits vrais" font partie de la relation : {(1,3), (2,6)}

- **Pièce (nop, nomp, prix)**

  domaine (nop) : N+, domaine (nomp) : chaînes de caractères, domaine (prix) : R+

Extension de relation

L'**extension** d'une relation est une instance de son schéma:

- C'est un ensemble de lignes correspondant au schéma à un instant t
- Représentée par un tableau à deux dimensions
  - chaque **colonne** correspond à un **attribut**
  - chaque **ligne** correspond à un **tuple**

✓Une relation est un ensemble de tuples et non une liste de tuples

- l'ordre des tuples n'a pas d'importance

Valeur NULL

`NULL` : valeur particulière indiquant que la valeur d'un attribut n'est pas connue pour un tuple ou que l'attribut ne s'applique pas

ex1. Cas un client dont on ignore la ddn

ex2. Cas d'un employé ne possédant pas de téléphone portable

NULL fait partie du domaine des attributs facultatifs (non obligatoires) d’une relation

Extension d'une relation Vin

Nom de la relation

Nom d'attribut

Vin

Cru	Millésime	Région	Couleur
Chenas	1983	Beaujolais	Rouge
Tokay	1980	Alsace	Blanc
Tavel	1986	Rhône	Rosé
Chablis	NULL	Bourgogne	Blanc
St-émilion	1987	Bordelais	Rouge

tuple

Types d’attributs

Il existe différents types d’**attribut** (*cf.* modélisation conceptuelle des données)

- ✓Attribut **atomique** / **composé**

  ex. *nom* versus *adresse* subdivisé en *rue, ville, CP*

- ✓Attribut **monovalué** / **multivalué**

  ex. *nom* versus *diplômes*

- ✓Attribut **dérivé** : calculé à partir d’autre(s) attribut(s)

  ex. *âge* calculé à partir *de date_de_naissance*

Contraintes d'intégrité : clé (primaire)

Clé d'une relation : ensemble minimal d'attributs qui identifient de manière unique tout tuple dans toute extension de la relation

cf. Notion d'identifiant de type d'entité dans modèle E-A
ex : {cru, millésime,couleur} dans la relation Vins
{no-sécu} dans une relation Personne.

Si une relation a plusieurs clés candidates, on en choisit une, c'est la **clé primaire**

ex. numProd : clé primaire de Produit.
{cru, millésime, couleur} : clé primaire de Vin.

Notation : la clé primaire est **soulignée** dans le schéma.

ex. Produit (<u>numProd</u>, libellé, pu)

Contraintes d'intégrité : clé étrangère

Clé étrangère d'une relation : ensemble d'attributs constituant la clé primaire d'une autre relation (ou dont la valeur est unique dans cette relation)

Les clés étrangères définissent les CI référentielles

Notation : la clé étrangère peut être représentée en **italique** dans le schéma

ex. Soit la base de données :

Buveur (nb,nom,prénom)

Vin (nv, cru, millésime,degré)

Abus (***nb,nv***,date,quantité)

Abus.nv **est une clé étrangère, elle référence** ***Vin.nv***

Abus.nb **est une clé étrangère, elle référence** ***Buveur.nb***

Contraintes d'intégrité spécifiques

Les données doivent vérifier certaines conditions pour être cohérentes

ex. `Date_début_projet < date_fin_projet`

`millésime > 1900`

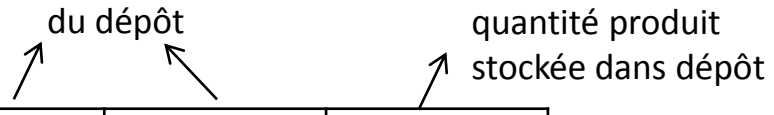
Plan du cours

1. Introduction
2. Modèle conceptuel de données Entité-Association
3. **Modèle relationnel de données**
    - Concepts du modèle relationnel
    - **Redondance des données et normalisation**
    - Passage du modèle entité-association au relationnel
    - Langages de manipulation des données (LMD)

4. Le langage SQL
5. Le langage PL/SQL
6. Transactions et concurrence d'accès

Exemple de schéma de relation redondante

Relation Produit



prod_id	libelle	pu	dep_id	adr	volume	qté
p1	DVD-R	23.5	d1	Nancy	9000	300
p1	DVD_R	23.5	d2	Laxou	6000	500
p3	CD-ROM	10	d4	Nancy	2000	900

Redondance :

- Ex: ***libellé*** et ***pu*** apparaissent pour tous les tuples relatifs au même produit

Risque d'introduction d'incohérence :

- Ex: lors de l'insertion d'une nouveau tuple relatif au produit ***p1,*** lors d'une mise à jour du ***libellé*** d'un produit

Risque de perte d'information :

- Ex: en cas de suppression du 3ème tuple, on perd le ***libellé*** et le ***pu*** de P3

Comment construire des schémas de relations normalisés ?

Une relation est ***normalisée*** si elle ne pose pas de problème de redondance ni de risques d'anomalies lors des mises à jour

Deux solutions existent:

1. Décomposer les relations redondantes en plusieurs relations normalisées
   a) Trouver les dépendances fonctionnelles (contraintes d'intégrité)
   b) Normaliser la ou les relations redondantes
2. Construire un modèle conceptuel de données (entité-association) puis transformer ce modèle en relations
   - Normaliser les relations qui le sont pas (s'il y en a)

Les dépendances fonctionnelles (DF)

Un ensemble d'attributs *Y dépend fonctionnellement* d'un ensemble d'attributs *X* dans une relation *R*, si étant donnée une valeur de *X*, il ne lui est associé qu'une seule valeur de *Y* dans toute extension de *R*

On note $X \xrightarrow{R} Y$ une telle dépendance fonctionnelle (DF)

On dit aussi :

- ✓ X détermine Y (ou Y est déterminé par X)
- ✓ le fait de connaitre X permet de connaitre Y
- ✓ Si deux tuples de R ont la même valeur de X alors ils ont la même valeur de Y quelle que soit l'extension de R

- ➢ Les DF sont un type important de **contraintes d'intégrité**
- ➢ Pour alléger l'écriture des DF, le nom de la relation au dessus de la flèche sera omis

DF dans la relation Produit

prod_id	libelle	pu	dep_id	adr	volume	qté
p1	DVD-R	23.5	d1	Nancy	9000	300
p1	DVD_R	23.5	d2	Laxou	6000	500
p3	CD-ROM	10	d4	Nancy	2000	900

Deux produits ne peuvent pas avoir le même numéro

prod_id → libellé

prod_id → pu

Deux dépôts ne peuvent pas avoir le même numéro

dep_id → adr

dep_id → volume

La quantité en stock ne dépend que du produit et du dépôt

prod_id, dep_id → qté

20

Propriétés des dépendances fonctionnelles (Axiomes d'Amstrong)

R est une relation; X, Y, W, Z sont des ensembles d'attributs de R

(F1) *Réflexivité :* $Y \subseteq X \Rightarrow X \rightarrow Y$ *(en particulier* $X \rightarrow X$*)*

(F2) *Augmentation :* $X \rightarrow Y \Rightarrow X \cup Z \rightarrow Y \cup Z$

(F3) *Union :* $X \rightarrow Y$ *et* $X \rightarrow Z \Rightarrow X \rightarrow Y \cup Z$

(F4) *Transitivité :* $X \rightarrow Y$ *et* $Y \rightarrow Z \Rightarrow X \rightarrow Z$

(F5) *Pseudo-transitivité :* $X \rightarrow Y$ *et* $Y \cup W \rightarrow Z \Rightarrow X \cup W \rightarrow Z$

(F6) *Décomposition :* $X \rightarrow Y$ *et* $Z \subseteq Y \Rightarrow X \rightarrow Z$

Clé d'une relation : autre définition

- Un ensemble d'attributs $X \subseteq U$ est une *clé* pour la relation R (U) si :
  (i) X $\rightarrow$ U
  (ii) X est m*inimal* (X est le plus petit ensemble d'attributs tel que X $\rightarrow$ U)

- Si une relation possède plusieurs clés *candidates*, nous on choisissons une qui sera appelée *clé primaire* (soulignée dans le schéma de la relation)

Comment trouver la clé d'une relation à l'aide des DF ?

Cela revient à chercher le plus petit ensemble d'attributs d'une relation qui détermine tous ses attributs

Soit la relation Véhicule de schéma

Véhicule (no_immat, no_châssis, modèle, marque, puissance)

Sachant que le n° d'immatriculation d'un véhicule est unique et qu'il en est de même pour le n° de châssis (n° VIN) :

1- Trouver les DF

2- Trouver les clés candidates de la relation Véhicule

23

Première forme normale (1NF)

Une relation est en ***1NF*** si chacun de ses attributs est atomique (non composé) et mono-valué

Personne (id, prénom, nom, diplômes)

où ***diplômes*** est l'ensemble des diplômes obtenus par une personne

La relation **Personne** n'est pas en 1NF

Comment normaliser en 1NF ?

Personne

id	prénom	nom	diplômes
p1	Jean	Maire	Licence Master Doctorat
p2	David	Tor	Licence Master

Possibilité 1

Personne1

id	prénom	nom	diplôme1	diplôme2	diplôme3
p1	Jean	Maire	Licence	Master	Doctorat
p2	David	Tor	Licence	Master	NULL

Possibilité 2

Personne2

id	prénom	nom
p1	Jean	Maire
p2	David	Tor

Diplômes

id	diplôme
p1	Licence
p1	Master
p1	Doctorat
p2	Licence
p2	Master

Personne1, Personne2, Diplômes : 1NF

- Avantages/inconvénients de chaque possibilité ?
- Quand privilégier chaque possibilité ?

Deuxième forme normale (2NF)

Une relation *R* munie d'une clé primaire est en ***2NF*** si

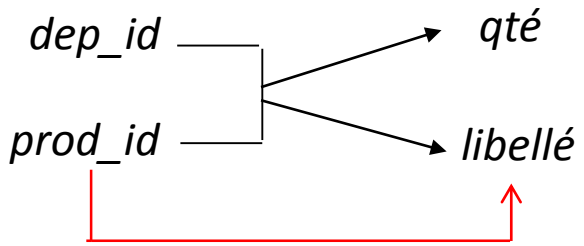
(i) elle est en 1NF et

(ii) tout attribut n'appartenant pas à la clé ne dépend pas d'une partie de la clé (DF partielle)

- **Stock1** (prod_id, dep_id, libellé, qté) n'est pas en 2NF car

  prod_id, dep_id $\rightarrow$ *qté, libellé*

  prod_id $\rightarrow$ *libellé*



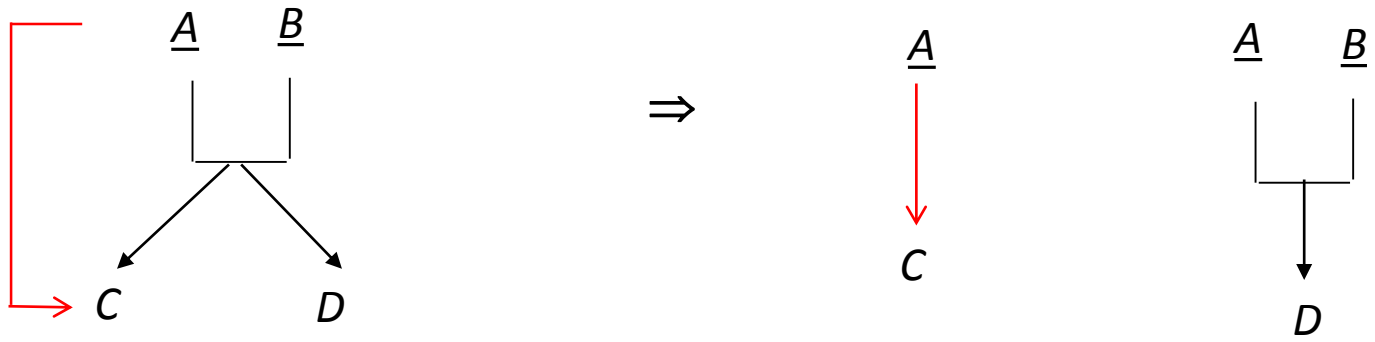
On peut décomposer **Stock1** en deux relations :

Stock (prod_id, dep_id, qté) relation 2NF

Produit (prod_id, libellé) relation en 2NF

Comment normaliser en 2NF ?

1. Isoler la DF qui pose problème dans une nouvelle relation
2. Supprimer la partie droite de la DF de la relation initiale



Troisième forme normale (3NF)

Une relation munie d'une clé primaire est en ***3NF*** si

(i) elle est en 2NF et

(ii) tout attribut n'appartenant pas à la clé ne dépend pas d'un autre attribut n'appartenant pas à la clé

Relation **Avion1**

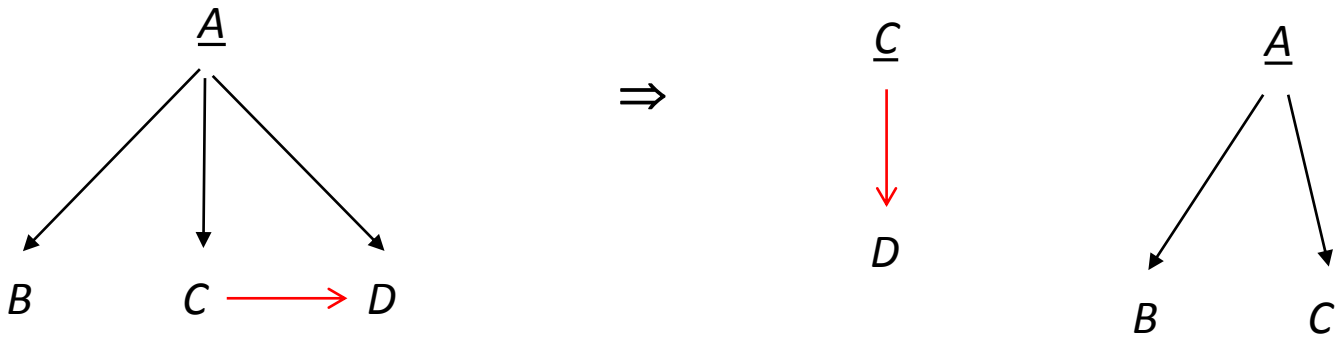
no_avion	constructeur	type	capacité	propriétaire
AH32	Boeing	B747	C1	Air France
FM34	Airbus	A320	C2	British Airways
BA45	Boeing	B747	C1	Egypt Air

type $\rightarrow$ *capacité* et *type* $\rightarrow$ *constructeur* et ces attributs ne sont pas dans la clé

La relation Avion1 est 2NF mais pas en 3NF. Comment normaliser ?

Comment normaliser en 3NF ?

1. Isoler la DF qui pose problème dans une nouvelle relation
2. Supprimer la partie droite de la DF de la relation initiale



Normalisation d'une relation : Théorème de Heath

- Toute relation a au moins une décomposition en 3NF qui est sans perte d'information* et qui préserve** les dépendances fonctionnelles (fermeture transitive)

- **Théorème de Heath :** Soit une relation **R(A,B,C)** telle que **A → B**
  - ✓ La décomposition (par projection) de R en deux relations **R1** (**A**,**B**) et **R2** (**A**,**C**) **préserve** la DF A → B et est **sans perte d'information**

    R est égale à la **jointure naturelle** R1 et R2 (sur le(s) attribut(s) commun(s) A)

Démarche naïve pour la normalisation d'une relation

1. Identifier les DF
2. Choisir une DF et appliquer le théorème de Heath pour décomposer la relation (qui n'est pas en 3NF)
3. Pour chaque relation issue de la décomposition qui n'est pas en 3NF, itérer à l'étape 2.

Le processus aboutit à une liste de relations en 3NF (décomposition sans perte d'information) mais certaines DF peuvent être perdues selon l'ordre du choix des DF

Plan du cours

1. Introduction
2. Modèle conceptuel de données Entité-Association
3. **Modèle relationnel de données**
    - Concepts du modèle relationnel
    - Redondance des données et normalisation
    - **Passage du modèle entité-association au relationnel**
    - Langages de manipulation des données (LMD)

4. Le langage SQL
5. Le langage PL/SQL
6. Transactions et concurrence d'accès

Passage du modèle entité-association au relationnel

Il y a une deuxième façon de construire un modèle relationnel normalisé :

1) Construire le modèle entité-association (E/A)

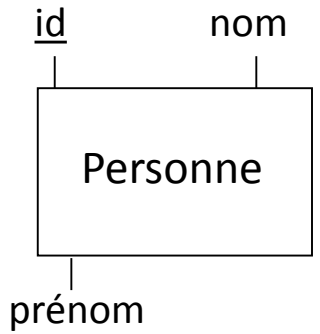
2) Transformer le modèle E/A en modèle relationnel (relations en 3NF) en plusieurs étapes

- à chaque **étape** on applique de façon **itérative** (à chaque élément du MCD) une **règle de transformation** dans **l'ordre** où elles sont données ici

3) Normaliser les relations qui ne le sont pas en utilisant les DF

Transformation d'une entité

- Transformer chaque entité E en une relation RE
- Les attributs de RE sont les attributs de E
- La *clé primaire* de RE est l'identifiant de E



⇨ Personne(id, nom, prénom)

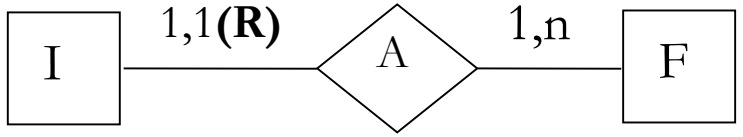
Modèle E/A (MCD)

Modèle relationnel (Modèle Logique de Données)

Cas d'une entité faible

- Chaque **entité faible** I, donne une relation R qui comprend
  - les attributs de I
  - l'identifiant (idf) de l'entité identifiante
  - les attributs de l'association identifiante (s'il y en a)

- La clé de R est la concaténation de l'identifiant relatif (idf_relatif) de I et de l'identifiant de l'entité identifiante *(c-à-d, la clé de la relation issue de cette entité)*



1. RF (Attr(F)...)
2. RI (Attr(I), Attr(A), clé(RF)...)

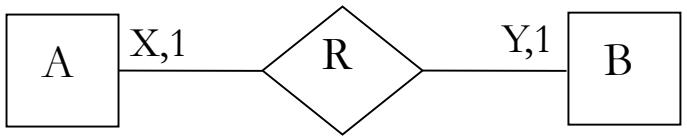
clé(RI)= idf_relatif (I) U idf (F)

Cas d’une association binaire 1-1

Chaque association binaire **1-1** est prise en compte en incluant la clé primaire d'une des relations comme clé étrangère dans l'autre relation

- idem pour les attributs éventuels de l'association

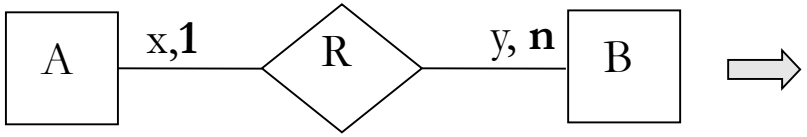
Fusion possible des deux relations en une relation si les cardinalités minimales sont non nulles



Cas d’une association binaire 1-N

Chaque association de type **1-N** est prise en compte en incluant la clé primaire de la relation de cardinalité N comme clé étrangère dans l'autre relation

- idem pour les attributs éventuels de l'association

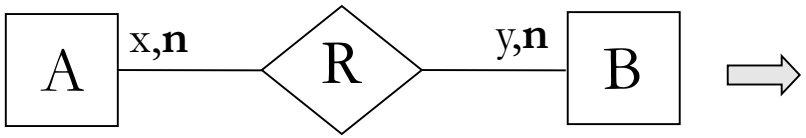


RA(Attr(A), **Attr(R), clé(RB)** ...)
RB(Attr(B)...)

Cas d'une association binaire N-M

- Chaque association de cardinalité **N-M** donne lieu à une nouvelle relation incluant :
  - les clés primaires des relations issues des entités participantes
  - les attributs de l'association

- C**hoix** d'une clé minimale parmi ces attributs (cf. Dépendances focntionnelles)

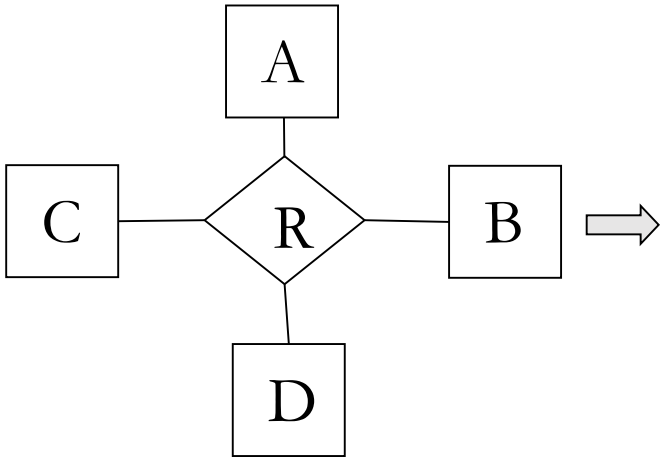


1. RA(Attr(A)...)
2. RB(Attr(B)...)
3. RR(clé(RA), clé(RB), Attr(R))

   Puis choix d'une clé pour RR

Cas d’une association n-aire (n>2)

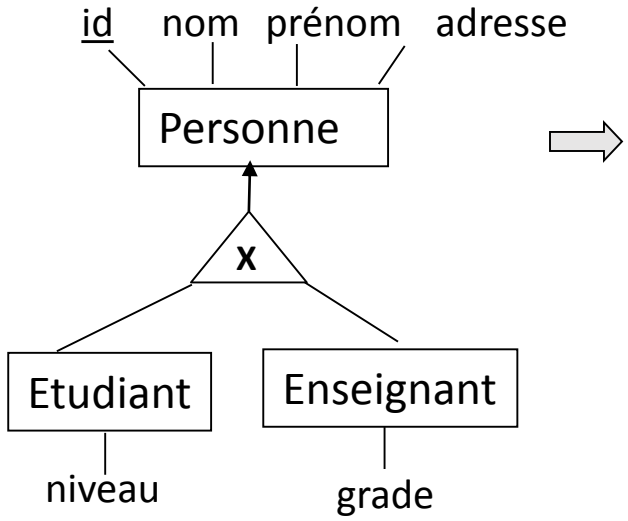
- Chaque association n-aire (n>2) donne lieu à une relation ayant comme attributs la liste des clés primaires des relations issues des entités participantes ainsi que la liste des attributs éventuels de l'association.
- Choix d'une clé minimale (cf. Dépendances fonctionnelles)



RA(Attr(A)...)
RB(Attr(B)...)
RC(Attr(C)...)
RD (Attr(D)...)
RR(clé(RA), clé(RB), clé (RC), clé(RD), Attr(R))
Puis choix d’une clé pour RR

Cas d'une spécialisation (règle générale)

- Créer une relation pour chaque entité générique et spécialisée
- Les attributs de chaque relation sont les attributs de l'entité correspondante + l'identifiant de l'entité générique pour les entités spécialisées
- La clé de chaque relation est l'identifiant de l'entité générique



Personne (<u>id</u>, nom, prenom, adresse)
Etudiant (<u>id</u>, niveau)
Enseignant (<u>id</u>, grade)

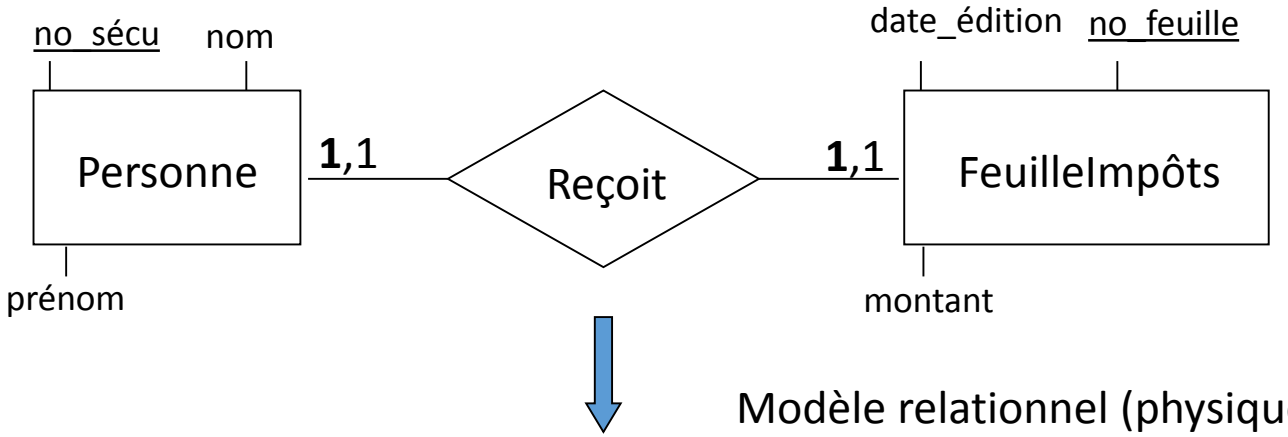
Quid d'une spécialisation XT (partition) ?

40

Cas d’un attribut calculé ou composé

- Chaque propriété calculée (attribut dérivé) donne lieu à une **vue** dont la définition correspond la règle de calcul (requête SQL)
  - ✓ Calculer âge à partir de la date du jour et de la DDN
  - ✓ Calculer le montant total (TTC) d’une commande
- Ne garder que les attributs atomiques formant un attribut composé à moins de vouloir le traiter comme un attribut atomique (ex. adresse)

Exemple 1
Association binaire un à un (zero cardinalité min à 0)



Modèle relationnel (logique)

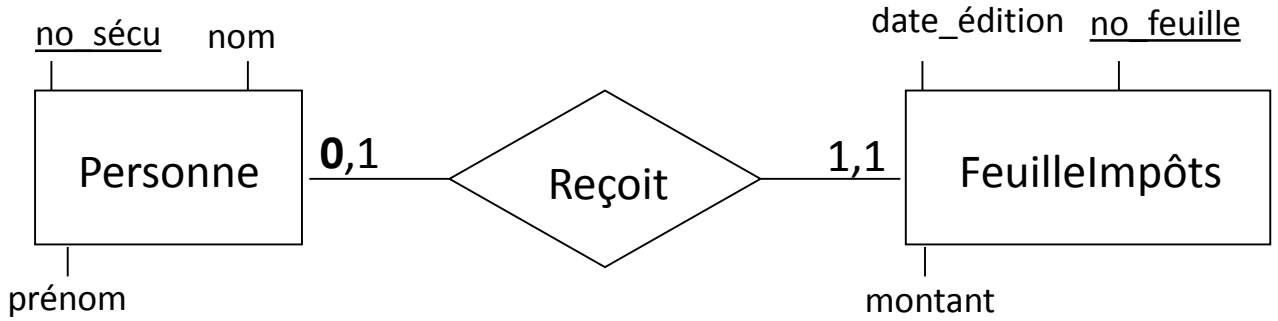
Personne_Impôts (no_sécu, nom, prénom, no_feuille, date_édition, montant)

CI : no_feuille est un attribut **unique** et **obligatoire**

Modèle relationnel (physique, SGBD Oracle)

```
CREATE TABLE Personne_Impôts
        (no_sécu number(15),
        nom varchar(100)
        prénom varchar(100),
        no_feuille number(15) NOT NULL,
        date_édition date,
        montant number(20,4),
        PRIMARY KEY (no_sécu),
        UNIQUE (no_feuille))
```

Exemple 2
Association binaire 1-1 (une cardinalité minimale à 0)



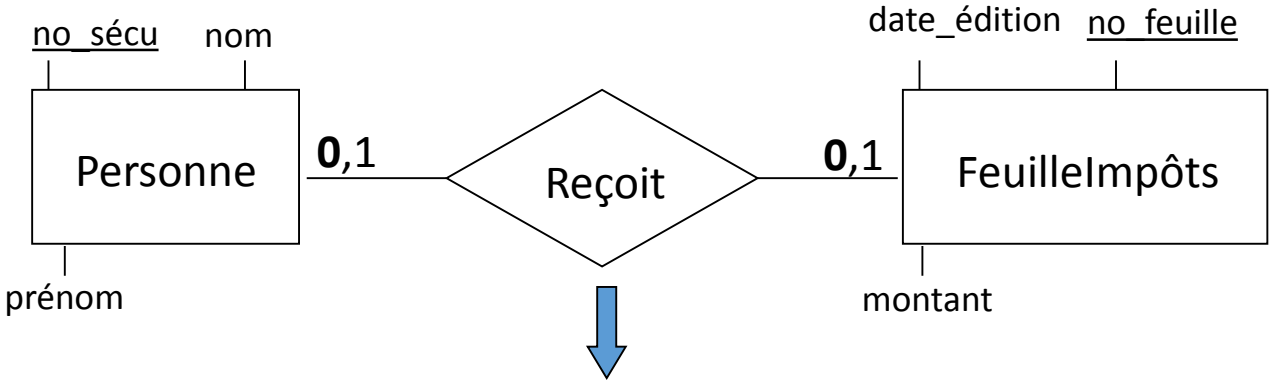
Contrainte d'intégrité (CI) supplémentaire : La suppression d'une personne doit entraîner la suppression de sa feuille d'impôts

Personne (no_sécu, nom, prénom)
FeuilleImpôts (no_feuille, date_édition, montant, ***no_sécu***)

FeuilleImpôts.no_sécu : attribut à valeur **obligatoire**
+ CI

```
CREATE TABLE Personne
   (no_sécu number(15),
   nom varchar(100),
   prénom varchar(100),
   PRIMARY KEY (no_sécu))
CREATE TABLE Feuille_Impôts
   (no_feuille number(15),
   date_édition date, montant number(20,4),
   no_sécu number(15) NOT NULL,
   PRIMARY KEY (no_feuille),
   FOREIGN KEY (no_sécu) REFERENCES
   Personne ON DELETE CASCADE)
```

Exemple 3
Association binaire 1-1 (cardinalités minimales à 0)



1ère possibilité :
Personne (no_sécu, nom, prénom)
FeuilleImpôts(no_feuille, date_édition, montant, ***no_sécu***)

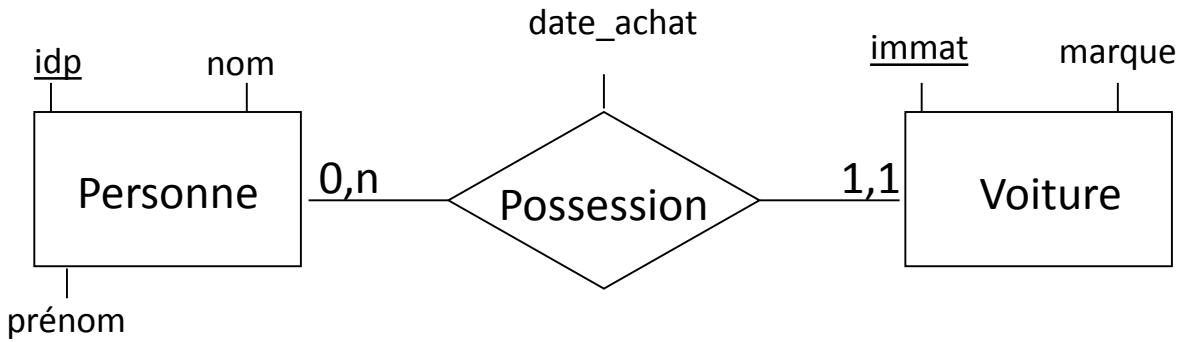
FeuilleImpôts.no_sécu : attribut à valeur **facultative**

2ème possibilité :
Personne (no_sécu, nom, prénom, ***no_feuille***)
FeuilleImpôts(no_feuille, date_édition, montant)

Personne.no_feuille : attribut à valeur **faculative**

```
CREATE TABLE Feuille_Impôts
 (no_feuille number(15),
 date_édition date, montant
 number(20,4),
 no_sécu number(15) NULL,
 PRIMARY KEY (no_feuille),
 FOREIGN KEY (no_sécu)
 REFERENCES
 Personne ON DELETE SET NULL)
```

Exemple 4
Association binaire 1-N



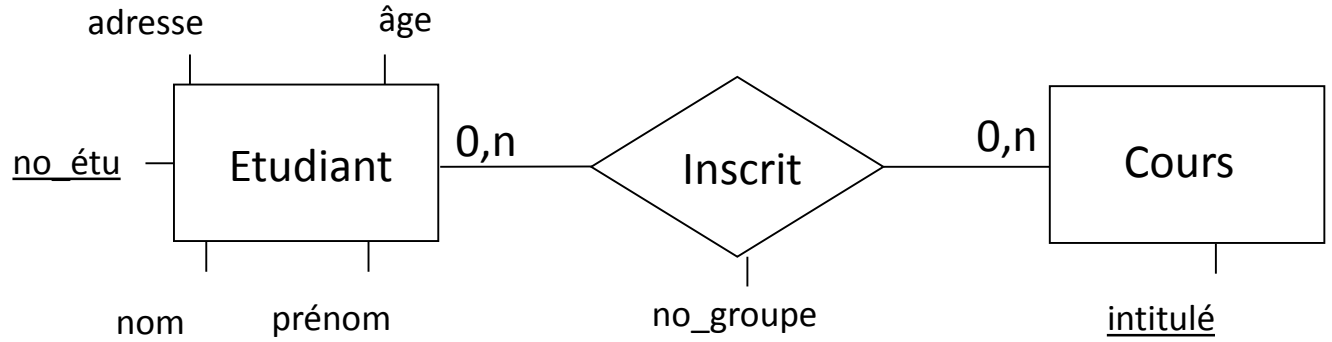
CI : La suppression d'une personne doit entraîner la suppression de ses voitures

Personne (idp, nom, prénom)
Voiture (immat, marque, ***idp,*** date_achat)

Voiture.idp : attribut à valeur **obligatoire**
+ CI

```
CREATE TABLE Voiture
 (immat varchar(10),
 marque varchar(100),
 idp number(20) NOT NULL,
 date_achat date ,
 PRIMARY KEY (immat),
 FOREIGN KEY (idp) REFERENCES
 Personne ON DELETE CASCADE  )
```

Exemple 5
Association N-M



CI

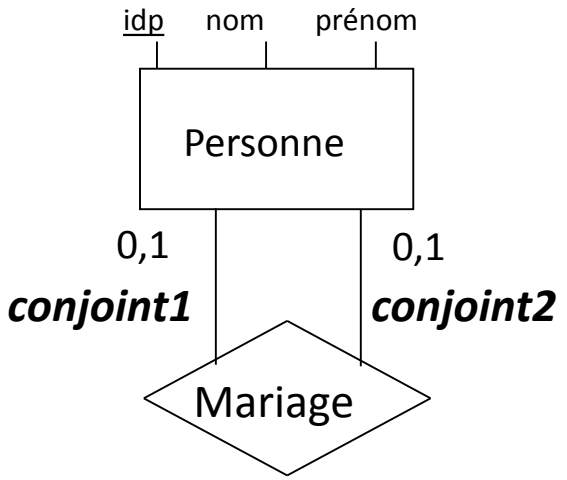
- no_étu, intitulé → no_groupe
- La suppression d'un étudiant doit entraîner la suppression de son inscription
- On ne peut supprimer un cours pour lequel il y a des inscrits

Etudiant (no_étu, nom, prénom, age, adresse)
Cours (intitulé)
Inscrit (*no_étu, intitulé*, no_groupe)

+ Deux dernières CI

```
CREATE TABLE Inscrit
 (no_étu  number(20) ,
 intitulé varchar(100),
 no_groupe number(2),
 PRIMARY KEY (no_étu,intitulé),
 FOREIGN KEY (no_étu) REFERENCES
 Etudiant ON DELETE CASCADE ,
 FOREIGN KEY (intitulé) REFERENCES
 Cours ON DELETE NO ACTION)
```

Exemple 6
association reflexive 1-1



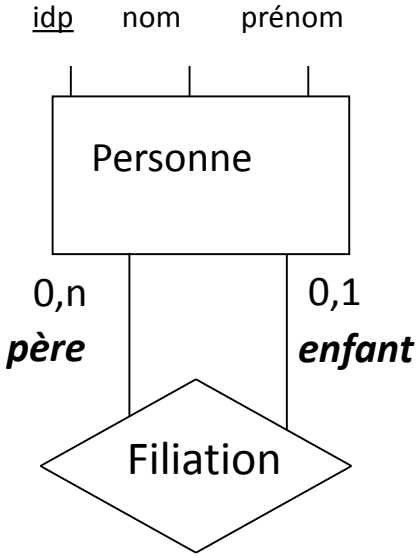
1ère possibilité :
Personne (idp, nom, prénom, *id_conjoint1*)

id_conjoint1 référence idp
id_conjoint1 attribut facultatif

```
CREATE TABLE Personne
  (idp number(20) ,
  nom varchar(100),
  prénom varchar(100),
  id_conjoint1 number(20) NULL,
  PRIMARY KEY (idp),
  FOREIGN KEY (id_conjoint1) REFERENCES
              Personne (idp) ON DELETE SET NULL)
```

2ème possibilité :
Personne (idp, nom, prénom, *id_conjoint2*)

Exemple 7
association réflexive 1-N

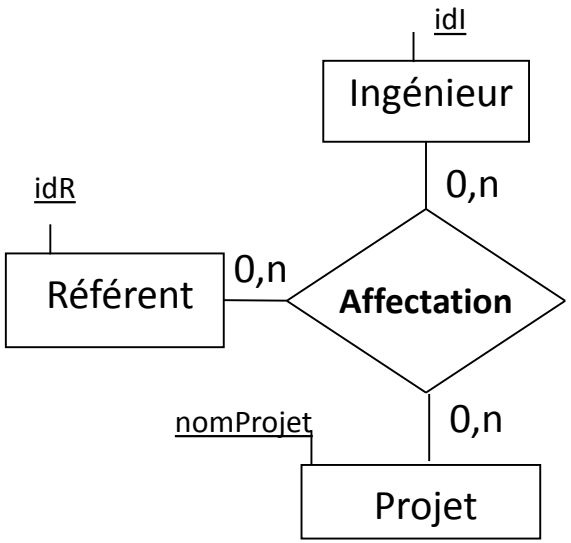


Personne (<u>idp</u>, nom, prénom, *id_père*)

id_père référence idp
id_père à valeur facultative

```
CREATE TABLE Personne
 (idp number(20) ,
 nom varchar(100),
 prénom varchar(100),
 id_père number(20) NULL,
 PRIMARY KEY (idp),
 FOREIGN KEY (id_père) REFERENCES
 Personne ON DELETE SET NULL)
```

Exemple 8
association ternaire (arité = 3)



CI

- Un ingénieur travaillant dans un projet a exactement un référent
- Il peut avoir plusieurs référents dans différents projets
- Un référent peut intervenir dans plusieurs projets

- DF : idI, nomProjet $\rightarrow$ idR
- Suppression d'une affectation en cas de suppression de l'ingénieur, du référent ou du projet concerné.

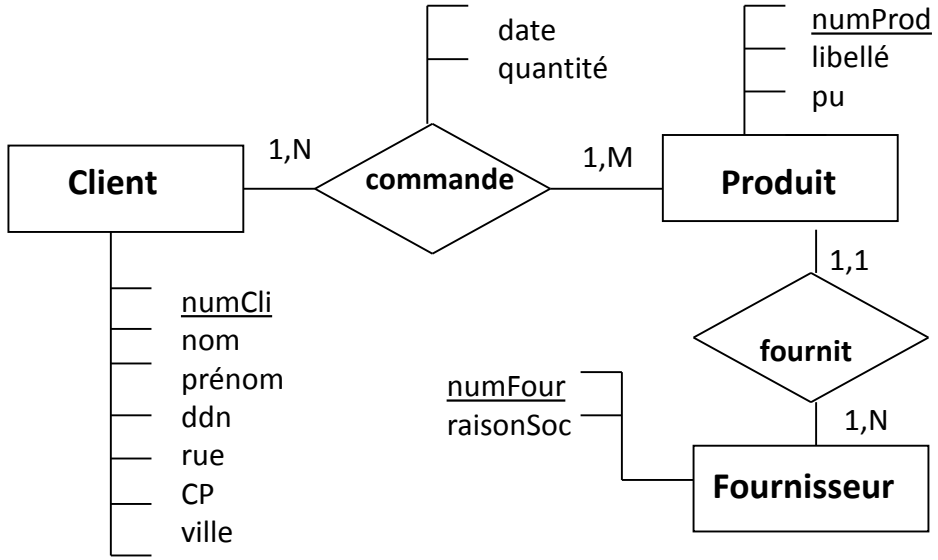
Quatre relations :

Référent (idR)
Ingénieur (idI)
Projet(nomProjet)
Affectation(*idI, nomProjet, idR*)

3 clés étrangères : Affectation.idI , Affectation.idR, Affectation.nomProjet

Exemple complet de transformation de l'exemple VPC

Modèle Entité-Association



Modèle relationnel

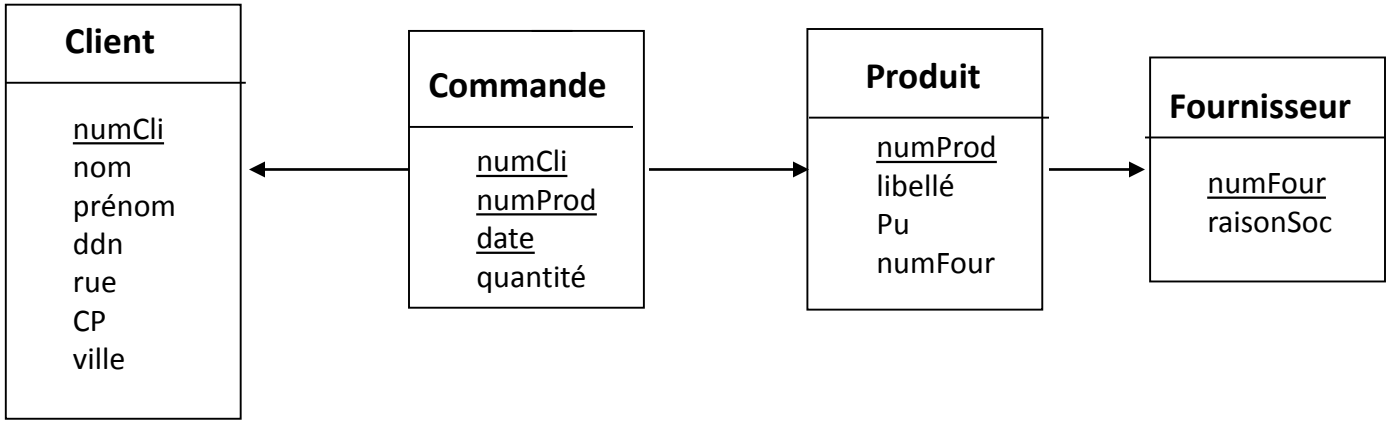
Client (numCli, nom, prénom, ddn, rue, CP, ville)

Produit (numProd, libellé, pu, ***numFour***)

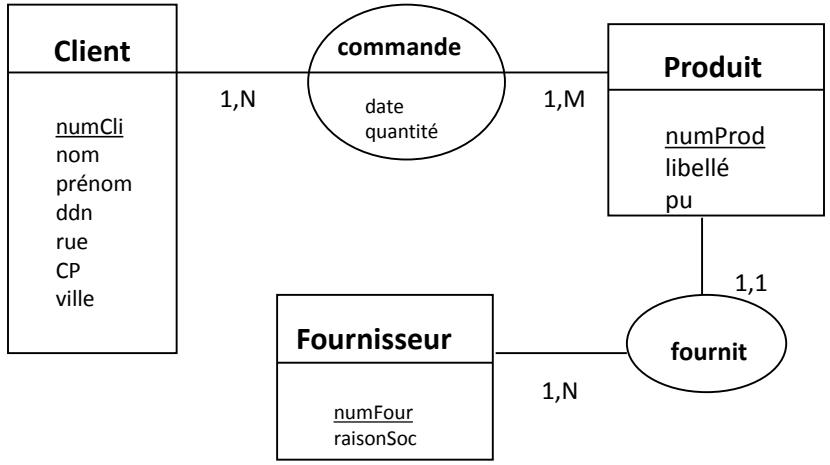
Fournisseur (numFour, raisonSoc)

Commande (***numCli***, ***numProd***, **date**, quantité)

Représentation **graphique** du **modèle relationnel** de données



à ne pas confondre avec le modèle conceptuel Entité-Association !



Plan du cours

1. Introduction
2. Modèle conceptuel de données Entité-Association
3. **Modèle relationnel de données**
    - Concepts du modèle relationnel
    - Redondance des données et normalisation
    - Passage du modèle entité-association au relationnel
    - **Langages de manipulation des données (LMD)**
        - **a) Algèbre relationnelle**
        - **b) Calcul Relationnel de Tuples (CRT)**

a) Algèbre relationnelle (A.R.)

A.R. : Langage de manipulation de données relationnelles

E. CODD (1970)

A.R. : huit opérateurs s'appliquant à une ou deux relations et donnant une relation comme résultat

- Union, intersection, différence
- Restriction (sélection)
- Projection
- Produit cartésien
- Jointure
- Division

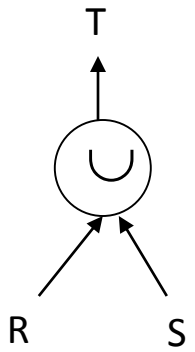
Opérateur d'union

Définition

L'**union** de deux relations R et S de même schéma est une relation T de même schéma contenant l'ensemble des tuples de R et S.

Notation

$T = UNION(R,S) = R \cup S$



Exemple d'union de relations

VINS-1

Numéro	Cru	Millésime	Degré
100	Chablis	1974	12
110	Mecurey	1978	13
120	Mâcon	1977	12

VINS-2

Numéro	Cru	Millésime	Degré
100	Chablis	1974	12
200	Sancerre	1979	11

UNION (VINS-1,VINS-2)

Numéro	Cru	Millésime	Degré
100	Chablis	1974	12
110	Mecurey	1978	13
120	Mâcon	1977	12
200	Sancerre	1979	11

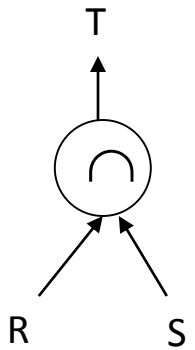
Opérateur d'intersection

Définition

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

Notation

T = INTERSECT (R,S) = R ∩ S



Exemple d'intersection de relations

VINS-1

Numéro	Cru	Millésime	Degré
100	Chablis	1974	12
110	Mecurey	1978	13
120	Mâcon	1977	12

VINS-2

Numéro	Cru	Millésime	Degré
100	Chablis	1974	12
200	Sancerre	1979	11

INTERSECT(VINS-1,VINS-2)

Numéro	Cru	Millésime	Degré
100	Chablis	1974	12

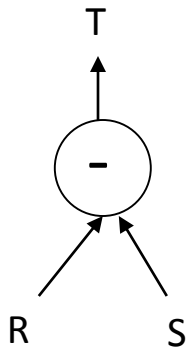
Opérateur de différence

Définition

La **différence** de 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.

Notation

T = DIFFERENCE (R,S) = R - S



Exemple de différence de relations

VINS-1

Numéro	Cru	Millésime	Degré
100	Chablis	1974	12
110	Mecurey	1978	13
120	Mâcon	1977	12

VINS-2

Numéro	Cru	Millésime	Degré
100	Chablis	1974	12
200	Sancerre	1979	11

MINUS (VINS-1, VINS-2)

Numéro	Cru	Millésime	Degré
110	Mecurey	1978	13
120	Mâcon	1977	12

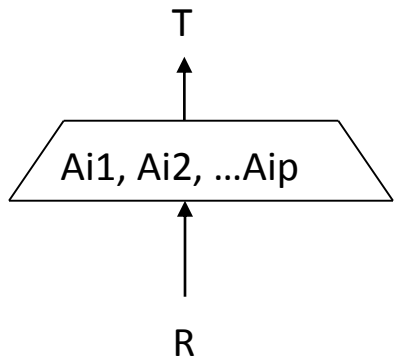
Opérateur de projection

Définition

La **projection** d'une relation R de schéma R(A1, A2,...,An) sur les attributs {Ai1, Ai2, ... , Aip} est une relation R' de schéma R' (Ai1, Ai2, ..., Aip) dont les tuples sont obtenus par élimination des attributs de R n'appartenant pas à R' et par suppression des tuples en double.

Notation

T = PROJECT (R ; Ai1, Ai2, ..., Aip)



Exemple de projection de relation

VINS

Numéro	Cru	Millésime	Degré
100	Chablis	1974	12
110	Mecurey	1978	13
120	Mâcon	1977	12
200	Sancerre	1977	12

PROJECT (VINS ; Millésime, Degré)

Millésime	Degré
1974	12
1978	13
1977	12

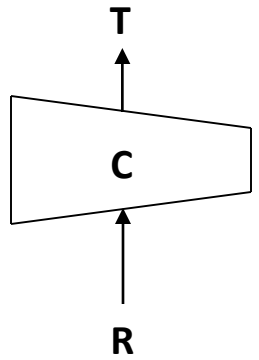
Opérateur de restriction (ou de sélection)

Définition

La restriction d'une relation R à l'aide d'une condition C est une relation R' de même schéma dont les tuples sont ceux de R qui satisfont la condition C.

Notation

T = RESTRICT (R ; C)



Exemple de restriction de relation

VINS

Numéro	Cru	Millésime	Degré
100	Chablis	1974	12
110	Mecurey	1978	13
120	Mâcon	1977	12
200	Sancerre	1977	12

RESTRICT (VINS ; Degré = 12)

Numéro	Cru	Millésime	Degré
100	Chablis	1974	12
120	Mâcon	1977	12
200	Sancerre	1977	12

Condition de restriction (sélection)

La condition C d'une restriction est une formule logique quelconque avec des connecteurs **ET** ($\wedge$) et **OU** ($\vee$) entre conditions simples de la forme $\mathbf{A_i}\ \theta\ \mathbf{a}$

où

- $\mathbf{A_i}$ est un nom d'attribut
- **a** est un élément du domaine de $\mathbf{A_i}$ (constante)
- θ est un des opérateurs **=, <, >, ≠, >=, <=**

ex. de condition de restriction :

```
(Cru="Chablis" ∨ Cru="Mâcon")∧ Millésime < 1988
```

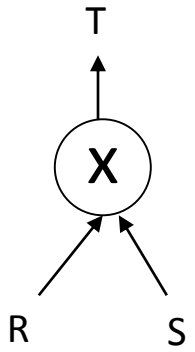
Produit cartésien

Définition

Le produit cartésien de deux relations R et S (de schémas quelconques) est une relation T ayant pour attributs la concaténation de ceux de R et de S et dont les tuples sont toutes les concaténations d'un tuple de R à un tuple de S (renommage des attributs de même nom).

Notation

T = PRODUCT (R,S) = R X S



Exemple de produit cartésien de relations

VINS-2

Numéro	Cru	Millésime	Degré
100	Chablis	1974	12
200	Sancerre	1979	11

VITICULTEURS

Nom	Ville	Région
Nicolas	Pouilly	Bourgogne
Martin	Bordeaux	Bordelais

VIGNOBLE = PRODUCT (VINS-2, VITICULTEURS)

Numéro	Cru	Millésime	Degré	Nom	Ville	Région
100	Chablis	1974	12	Nicolas	Pouilly	Bourgogne
100	Chablis	1974	12	Martin	Bordeaux	Bordelais
200	Sancerre	1979	11	Nicolas	Pouilly	Bourgogne
200	Sancerre	1979	11	Martin	Bordeaux	Bordelais

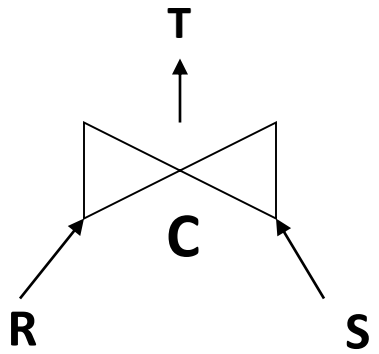
Opérateur de jointure

Définition

La jointure de deux relations R et S selon une condition C est l'ensemble des tuples du produit cartésien R X S satisfaisant la condition C.

Notation

T = JOIN (R, S ; C)



N.B. JOIN(R,S ; C) = RESTRICT(PRODUCT(R,S) ; C)

Exemple de jointure de relations

VINS

Numéro	Cru	Millésime	Degré
100	Chablis	1974	12
110	Mecurey	1978	13
120	Mâcon	1977	12

VITICULTEURS

Nom	Ville	Région
Nicolas	Pouilly	Bourgogne
Félix	Mâcon	Bourgogne

JOIN (VINS, VITICULTEURS ; Cru = Ville)

Numéro	Cru	Millésime	Degré	Nom	Ville	Région
120	Mâcon	1977	12	Félix	Mâcon	Bourgogne

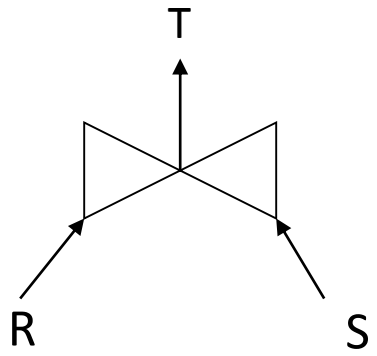
Jointure naturelle

Définition

La jointure naturelle de deux relations R et S est l 'équi-jointure (opérateur d'égalité) de R et S sur tous leurs attributs communs.

Notation

T = JOIN (R, S)



N.B. La condition de jointure implicite : égalité des paires d'attributs communs à R et S

Exemple de jointure naturelle de relations

VINS

Numéro	Cru	Millésime	Degré
150	Riesling	1984	11
110	Mecurey	1978	13
120	Mâcon	1977	12

VITIC

Nom	Numéro	Région
Nicolas	150	Alsace
Félix	120	Bourgogne

JOIN (VINS, VITIC)

Numéro	Cru	Millésime	Degré	Nom	Région
150	Riesling	1984	11	Nicolas	Alsace
120	Mâcon	1977	12	Félix	Bourgogne

Jointure externe

Jointure externe : inclure les tuples « célibataires » dans la jointure

- Jointure externe à gauche

  LEFT-JOIN(R,S) (jointure de R et S + «célibataires» de R)

- Jointure externe à droite

  RIGHT-JOIN(R,S) (jointure de R et S + «célibataires» de S)

- Jointure externe pleine

  FULL-JOIN(R,S)(jointure + «célibataires» de R et de S)

Exemple de jointure externe à gauche :

chercher les informations sur les vins et leur viticulteur en incluant les vins qui n'ont pas de viticulteur

VINS

Numéro	Cru	Millésime	Degré
150	Riesling	1984	11
110	Mecurey	1978	13
120	Mâcon	1977	12

VITIC

Nom	Numéro	Région
Nicolas	150	Alsace
Félix	120	Bourgogne

LEFT-JOIN(VINS , VITIC)

Numéro	Cru	Millésime	Degré	Nom	Région
150	Riesling	1984	11	Nicolas	Alsace
110	**Mecurey**	**1978**	**13**	**null**	**null**
120	Mâcon	1977	12	Félix	Bourgogne

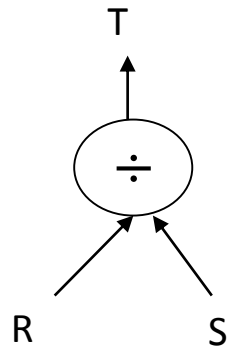
Opérateur de division

Définition

La division d'une relation R(X,Y) par une relation S(Y) est la projection de R sur X restreinte aux tuples en liaison avec **tous** les tuples de S.

Notation

$T = DIV(R,S) = R \div S$



Définition formelle

$R(X,Y) \div S(Y) = T(X) = \{ <x> \mid \forall\ y, <y> \in S \Rightarrow <x,y> \in R \}$

Exemple de division de relations

Produit

numProd	libellé	pu
P1	K7	5.5
P2	Vis	0.3
P3	Ecrou	0.4

Stock

numProd	numDep	qté
P1	D1	1000
P1	D2	-100
P1	D4	1200
P2	D1	-400
P2	D2	2000
P2	D4	1500
P3	D1	3000
P3	D4	2000

Numéro des dépôts stockant tous les produits :

DIV (PROJECT(Stock ; numProd, numDep),
PROJECT(Produit ; nomProd))

numDep
D1
D4

Exemple de requête algébrique (1/2)

Client (numCli, nom, prénom, ddn, rue, CP, ville)

Produit (numProd, libellé, pu, ***numFour***)

Fournisseur (NumFour, raisonSoc)

Commande (***numCli***, ***numProd***, date, quantité)

Ex. de question : Donner les produits commandés en quantité supérieure à 100 et dont le prix dépasse 1000€ . On affichera les numéros de produit, leur libellé et leur prix unitaire ainsi que la date de la commande.

Une requête algébrique = composition d'opérateurs algébriques

Notons **Res** la relation résultat (R_1, R_2, R_3 : relations intermédiaires)

Exemple de requête algébrique (2/2)

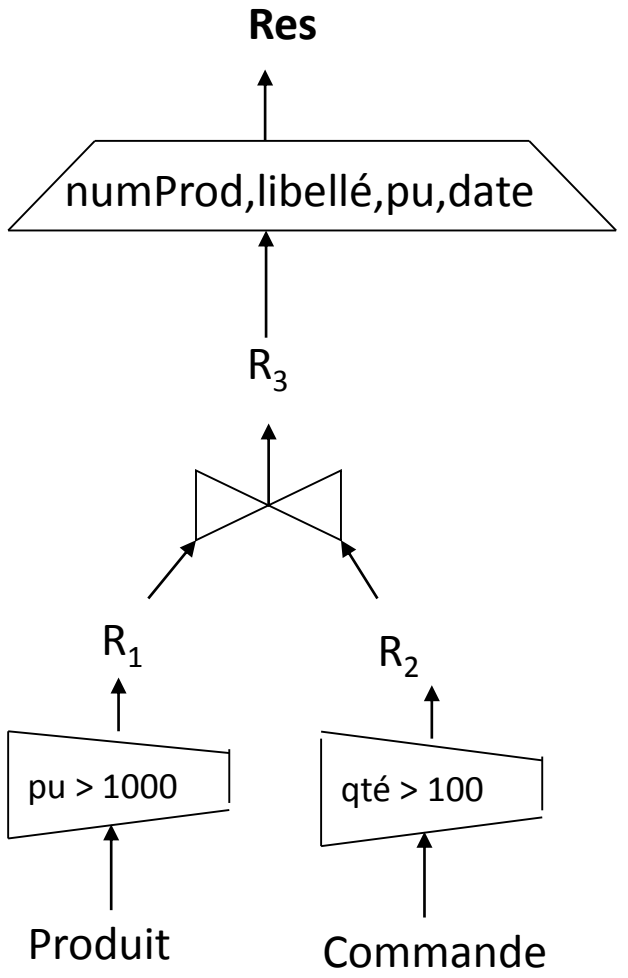
R_1 = RESTRICT (Produit ; pu>1000)

R_2 = RESTRICT (Commande ; qté >100)

R_3 = JOIN (R1, R2)

Res =

PROJECT (R3 ; numProd, libellé, pu, date)



A vous : construire les requêtes algébriques répondant aux questions (BD VPC)

1) Libellé et prix unitaire de tous les produits

2) Libellé des produits de prix inférieur à 50€

3) Nom et prénom des clients ayant commandé le produits numéro 56.

4) Nom des clients ayant commandé au moins un produit de prix supérieur à 500€.

5) Nom des clients n'ayant pas commandé le produit numéro 56.

A.R. : Ensemble minimal d’opérateurs

- Les requêtes (SQL) dans les SGBDR sont transformées en expressions algébriques

- Cinq opérateurs sont nécessaires (ensemble minimal)

  union, différence, projection, produit cartésien, sélection

- Les autres opérateurs s'expriment en fonction des précédents.

A.R. : Quelques propriétés des opérateurs

- RESTRICT (R ; C1 $\wedge$ C2) = RESTRICT (RESTRICT(R ; C1) ; C2)
- RESTRICT(RESTRICT(R ; C1) ; C2) = RESTRICT(RESTRICT(R ; C2) ; C1)
- PROJECT (PROJECT (R ; liste1) ; liste2) = PROJECT (R ; liste2)
- JOIN(R ,S ; C) = JOIN (S,R ; C)
- JOIN (JOIN(R,S ; C),T ; C') = JOIN (R, JOIN(S ,T ; C') ; C)

où R, S, T : noms de relation

C, C', C1, C1, C2 : conditions

liste1, liste2 : listes d'attributs

Propriétés utilisées pour l'**optimisation** de requêtes dans les SGBDR

b) Calcul Relationnel de Tuples (CRT)

CRT = Langage du 1er ordre (CP1) pour manipuler des données relationnelles

- même pouvoir d'expression que l'algèbre relationnelle

Exemple de requête : **{** $t.a_1, t.a_2 \ldots, t.a_n$ **|** $p(t)$ **}**

- Le **résultat** de la requête inclut tous les tuples **t** qui rendent la formule **p(t) vraie**

- La **formule** p(t) est définie récursivement en partant de **formules atomiques** et en construisant des formules de plus en plus complexes au moyen des **connecteurs logiques**.

80

Syntaxe d'un langage du 1er ordre

L = (A,F)

- A : Alphabet
- F : Formules (bien formées) construites sur A

A contient

- {Constantes} : a, b, c ...
- {Variables} : x, y, z ...
- {Fonctions} avec arité : f( ), g( ), h( ) ....
- {Prédicats}avec arité : P( ), Q( ), R( ) ....

Prédicat : fonction booléenne à 2 valeurs possibles Vrai/Faux

Syntaxe d'un langage du 1er ordre

- Définition d'un terme
  - Toute constante est un terme
  - Toute variable est un terme
  - Si **t1, t2, ..tn** sont des termes et si **f** est une fonction n-aire, alors **f(t1, t2, ..,tn)** est un terme
- Définition d'un atome
  - Si **t1, t2, ..., tn** sont des termes et si **P** est un prédicat n-aire alors **P(t1, t2, .., tn)** est un atome
- Définition d'une formule
  - Tout atome est une formule
  - Si **P** et **Q** sont deux formules alors

    $\neg P$, $P \vee Q$, $P \wedge Q$, $P \Rightarrow Q$, $P \Leftrightarrow Q$, $\forall x\, P(x)$, $\exists x\, P(x)$

    sont des formules

Sémantique d'un langage du 1er ordre

- Exemples de formules (Syntaxe)
  - **Grand-père** (Jean , Marie)
  - **Egal** (double(4) , 8)
  - $\forall$x **Nombre** (x) $\Rightarrow$ ($\exists$y **PlusGrandQue** (y, x)
  - $\forall$x, $\exists$y (**P**(x) $\wedge$ **Q**(y)) $\Rightarrow$ (**R**(a) $\vee$ **Q**(b))

- Sémantique : sens d'une formule (V/F)
  - Théorie de la preuve
  - Théorie du modèle : Tables de vérité
    - Quand on attribue des valeurs a chaque terme et à chaque symbole de prédicat dans une formule on dit qu'une interprétation est donnée à la formule ou qu'on l'évalue
      - Résultat de l'interprpétation d'une formule : Vrai ou Faux

CRT et langage du 1er ordre

CRT est un langage du 1er ordre en considérant que dans les formules :

- Les prédicats comportent :
  - Un **prédicat unaire pour chaque relation** (même nom)
  - Comparateurs logiques (=, ¬ =, >, >=…)
- Les **variables** sont associées à des **tuples** de relations
- Les termes sont :
  - Constantes associées aux éléments des domaines
  - Fonctions de projection d'une variable de relation sur un de ses attributs (p.nom, p.pu…)

*R*ésultat de la requête **{ $t.a_1$,..., $t.a_n$, $s.b_1$, ...$s.b_k$ | p(t,s) }**

- projections sur les attributs **a_1... a_n** des tuples **t** qui rendent vraie la formule **p(t,s)**
- projections sur les attributs **b_1... b_n** des tuples **s** qui rendent vraie la formule **p(t,s)**

CRT : Variables liées / variables libres

$\forall x (P(x) => Q(x, y))$

x est une variable **liée** et **y** est une variable **libre**.

Une formule ne peut être évaluée que lorsque toutes les variables sont liées

Dans une requête en CRT $\{t.a_1, t.a_2\ldots, t.a_n \mid p(t)\}$

la variable **t** qui apparaît à la gauche de `|' doit être la **seule** variable **libre** dans la formule p(t)

Exemples de requêtes dans le CRT

Soit la BDR de schéma :

Produit (numProd, libellé, pu)

Dépôt (numDep, capacité, adresse)

Stock (numProd, numDep, qté)

La formule **Produit (p)** signifie **p** est un tuple de la relation Produit

La constante **p.numProd** désigne le numéro du produit **p**

- Nom et prix unitaire de tous les produits

  { p.libellé, p.pu | Produit (p) }

- Numéro des produits stockés dans le dépôt 'D2'

  { p.numProd | Produit (p) $\wedge$ $\exists$s

  (Stock(s) $\wedge$ s.numProd=p.numProd $\wedge$ s.numDep = 'D2') }

- Adresse et numéro des dépôts stockant tous les produits

  { d.adr, d.numDep | Dépôt (d) $\wedge$ $\forall$p

  (Produit(p) $\Rightarrow$ $\exists$s (Stock(s) $\wedge$ p.numProd=s.numProd $\wedge$ s.numDep=d.numDep))}

Calcul Relationnel de Tuples : Bilan

- Le Calcul relationnel de tuples est non-opérationnel, tout comme l'Algèbre relationnelle

- On exprime dans les requêtes ce qu'on veut obtenir et non comment l'obtenir
  - C'est un langage **déclaratif**

- L'algèbre et le calcul relationnel de tuples ont le même pouvoir expressif (notion de complétude relationnelle)
  - Chaque requête exprimable en algèbre relationnelle l'est aussi en CRT et vice-versa

- SQL dérive de l'algèbre relationnelle et du CRT