Le Modèle relationnel

(Version 1)

2010

- **Concepts de base**
  - Éléments de théorie des ensembles
  - Notion de « Relation » ou « Table »
- **Conception de schémas relationnels : la normalisation**
  - Notion de « Dépendance Fonctionnelle »
  - Notion de clé
  - 1NF, 2NF, 3NF, BCNF

Le Modèle Relationnel

- **Version 1 - CODD en 1970** (IBM San José)
  - basé sur des **concepts simples** (populaire)
  - **base théorique solide** (théorie des ensembles, logique formelle)
- **Version 2 - CODD en 1990**
  - **l'objet** : surmonter les défauts de la manipulation des données du langage SQL
  - **nouveaux opérateurs** pour **fermer le langage SQL** : *Jointure récursive, Cadrage (ou framing)*
  - **autres extensions** : *l'"extension" , opérateurs ensemblistes "externes", la jointure externe, intégration de fonctions définies par l'utilisateur au langage SQL*
- **Version 3 - CODD en 1990**
  - l'objet **extension du langage SQL** et passage à **l'objet**
  - intégration de **fonctions** définies par **l'utilisateur** au modèle
  - la **jointure récursive** généralisée aux graphes cycliques et acycliques
  - **hiérarchie des types et des domaines** permettant d'envisager l'approche objet et de ses propriétés (encapsulation, héritage, ...)

Eléments de théorie des ensembles (1)

- Un **ensemble** est défini par une **propriété** qui le caractérise
- cette propriété permet de déterminer **l'appartenance ou non d'un élément à l'ensemble**

  Ex :

  soit l'ensemble Z, défini par la propriété "être un entier naturel"

  1 appartient à Z noté : $1 \in Z$; 0,2 n'appartient pas à Z noté : $0,2 \notin Z$
- **définition en extension** d'un ensemble :

  Ex :

  E = {a, b, c, d}
- **définition en intention** d'un ensemble par sa propriété, définie comme un prédicat p(x)

  Ex :

  soit N l'ensemble des entiers positifs :

  un élément n appartiendra à N ssi il vérifie le prédicat p(x) = "$x \in Z$ et $x \geq 0$"
- **inclusion** de 2 ensembles :

  Ex :

  E1 est inclus dans E2 ssi $\forall x \in E1$ alors $x \in E2$

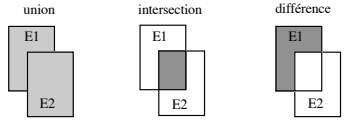
Eléments de théorie des ensembles (2)

- **opérateurs ensemblistes :**

  union : E1 ∪ E2 : $p_{E1}(x) \vee p_{E1}(x)$

  intersection : E1 ∩ E2 : $p_{E1}(x) \wedge p_{E1}(x)$

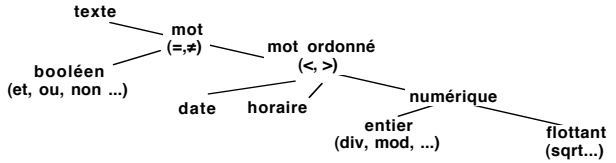
  différence : E1 – E2 : $p_{E1}(x) \wedge \neg p_{E1}(x)$



- **notion de domaine :**

  un domaine est un ensemble non vide dénombrable; exemples : N, {ens. des couleurs d'un feu de circulation}, ... semblable au type dans les langages de programmation

  typologie des domaines (Léonard 88) :



Eléments de théorie des ensembles (3)

Produit cartésien :

le produit cartésien d'un ensemble de domaines D1, D2,... Dn, noté D1 X D2 X ... Dn est l'ensemble des n-uplets ou Tuples <v1, v2,... vn> (avec vi ∈ Di)

Ex :

D1 = {vert, orange, rouge}

D2 = {0, 1}

D1 x D2 =

vert	0
vert	1
orange	0
orange	1
rouge	0
rouge	1

6 tuples

Notion de Relation (Table)

Etant donné n domaines D1, D2... Dn (non nécessairement distincts), on appelle **relation** (n-aire) sur ces n domaines, tout sous-ensemble du produit cartésien D1 x D2 x ... Dn

Ex : D1 {bleu, blanc, vert, rouge}, D2 {0, 1}, la relation MER suivante :

MER	D1	D2
	bleu	0
	bleu	1
	blanc	0
	vert	1

tuple d'une relation : <d1, d2... dn> appartenant à R; *ex. <bleu, 0>*

cardinalité d'une relation : nb de tuples appartenant à la relation; *cardinalité de MER = 4*

degré d'une relation : n, nombre de domaines impliqués (non nécessairement distincts); *degré de MER = 2*

attribut d'une relation : une colonne associée à un domaine caractérisée par un nom

Pour éviter la confusion entre le concept de « relation » du modèle Entité-Relation on parlera de « **Table** »

Notion de Table (Relation)

- **Schéma d'une table :** permet de définir une table, est constitué du nom de la table suivi de la liste de ses attributs avec leurs domaines de variation ainsi que de l'ensemble des contraintes d'intégrité associées à la table (définies plus loin)

  Ex : une table PIECE décrivant des pièces aura comme schéma :
  PIECE (n° : entier, nom : car (10), couleur : couleur, POIDS : réel, ville : car (10))

- **Extension d'une table :** Étant donnée une table, définie par son schéma, une extension de cette table sera un ensemble de lignes (rows) ou **tuples** définies par les valeurs prises par les attributs :

Ex : une extension de la table PIECE pourrait être :

PIECE	n°	nom	couleur	poids	ville
tuple 1	P1	écrou	rouge	14	Londres
tuple 2	P2	boulon	vert	17	Paris
tuple 3	P3	vis	bleu	12	Rome
tuple 4	P4	vis	rouge	14	Londres
tuple 5	P5	rivet	bleu	12	Paris
tuple 6	P6	clou	vert	10	Londres

cardinalité de la table PIECE = 6 (6 tuples); degré de la table PIECE = 5 (5 attributs)

Remarque : pour une extension de table, l'ordre des tuples ou l'ordre des attributs, ne sont pas significatifs.

Base de données relationnelle

- On définira une **base de données relationnelle** comme un ensemble de tables.
- Son **schéma** sera l'ensemble des schémas des tables la composant :

FOURNISSEUR	#F	nom_four	ville
	F1	Dupont	Paris
	F2	Bergeron	Quebec
	F3	Thomas	Geneve

FOURNIT	#F	#P	délais
	F1	P1	20
	F1	P3	10
	F2	P1	20
	F2	P4	30

PIECE	#P	nom_pièce	dépôt
	P1	table	Québec
	P2	fenêtre	Genève
	P3	chaise	Montréal
	P4	porte	Paris

Sans préciser les domaines de chacun des attributs, les **schémas de cette BD** est :

Table FOURNISSEUR (#F, nom_four, ville) *entité de l'E-R*
Table PIECE (#P, nom_pièce, dépôt) *entité de l'E-R*
Table FOURNIT (#F,#P, délai) *relation de l'E-R*

Les contraintes d'intégrité (introduction)

Contrainte d'intégrité = **assertion** (condition) qui doit être **vérifiée** par les valeurs d'attributs de tables constituant une BD

- **Contrainte d'unicité de valeur : clé primaire d'une table**
  - permet d'identifier de façon unique chaque tuple d'une table, on distinguera un ou plusieurs attributs dont les valeurs prises seront déclarées uniques pour toute extension de cette table : **clé primaire simple** (un seul attribut) ou **clé primaire composée** (plusieurs attributs) :

    Ex :
    - **table FOURNISSEUR (#F, nom_four, ville) ;** ***(#F) = clé primaire simple***
    - **table PIECE (#P, nom_pièce, dépôt) ;** ***(#F) = clé primaire simple***
    - **table FOURNIT (#F,#P, délai) ;** ***(#F,#P) = clé primaire composée***

    clé primaire équivalente à propriété identifiante dans l'Entité-Relation

- **Contrainte référentielle : clé étrangère d'une table**
  - traduit un lien sémantique entre 2 table A et B, réalisé par une duplication de la clé primaire de la table A dans la table B = **clé étrangère** :

    Ex :
    - **table FOURNIT (#F,#P, délai) : #F et #P sont chacune des clés étrangères provenant des tables FOURNISSEUR et PIECE.**

Conception de schémas relationnels

2 approches :

- **Approche par décomposition :**

  → théorie de la normalisation : Formes normales

OU

- **Approche par modélisation conceptuelle :**

  1. Modélisation Conceptuelle des Données (MCD) en formalisme Entité-Relation
  2. Dérivation d'une Modélisation Logique des Données Relationnelle (MLDR)
  3. Dérivation des schémas relationnels

Problèmes liés à la conception de schémas relationnels

Soit une extension de la table P concernant des propriétaires de véhicules :

P.	nom	date	tel	n°immat	marque	type	cv	coul.
tuple1	Durand	10/2/88	422775	540HB75	*renault*	*R25S*	9	bleu
tuple2	Dupont	8/10/88	665241	301UP75	peugeot	405GR	7	vert
tuple3	*Pagnol*	7/7/89	*912458*	741XD13	citroën	AX11	4	blanc
tuple4	*Pagnol*	21/4/90	*912458*	848HP13	volvo	245	8	gris
tuple5	Duval	15/8/90	425896	120DR75	citroen	BX17	6	blanc
tuple6	Martin	10/7/90	522684	956BV42	*renault*	*R25S*	9	rouge

- **données redondandes :**
  - une personne apparaît autant de fois qu'elle possède de voiture→ *risque d'incohérences (changement de # de tél.)*
  - redondance *type→ marque*
- **nécessité de valeur nulles :** voiture sans propriétaire ? propriétaire sans voiture ?

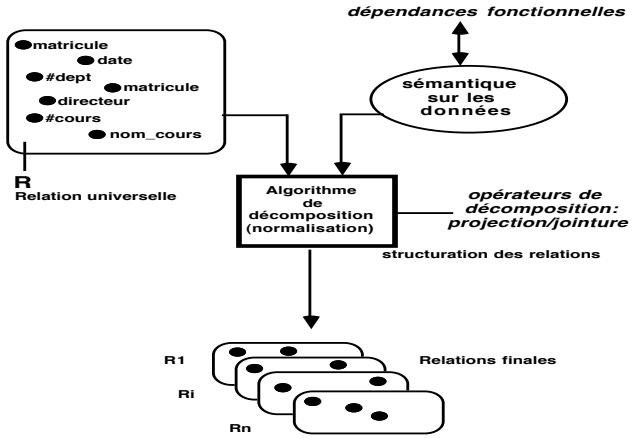
Pourquoi :

- une mauvaise conception des entités et des associations représentant le réel conduit à des tables problématiques.
- une table qui ne représente pas de vraies entités ou associations présente des incohérences potentielles et nécessite un codage des valeurs nulles

→ **Concevant de bons schémas de tables : normalisation des tables**

Conception de schémas relationnels : Approche par la normalisation

(Ullman 80)

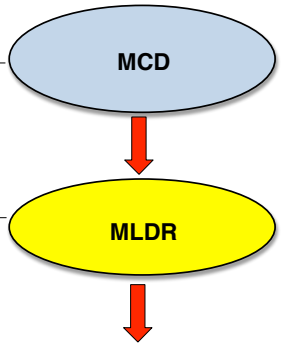


Conception de schémas : Approche par modélisation conceptuelle

NIVEAU CONCEPTUEL

MCD (Modèle Conceptuel de Données)

Modèle graphique en Formalisme Entité-



NIVEAU LOGIQUE

MLDR (Modèle Logique de Données Relationnel)

Modèle graphique en Formalisme Merise

Schémas relationnels

(par simple lecture)

Avantages:

- progressive, cognitive
- graphique -> dialogue
- permet une validation par modèles externes
- présente dans de nombreuses méthodologies de génie logiciel

Décomposition/Recomposition des tables

2 opérations élémentaires inverses :

- **Projection :** consiste à supprimer des attributs d'une table et à éliminer les tuples en double de la nouvelle table obtenue:

Soit $R(A_1, A_2,..., A_n)$; $\{A_{i1}, A_{i2},..., A_{ip}\} C \{A_1, A_2,..., A_n\}$ avec $(ij \neq ik)$

$$\Pi_{Ai1, Ai2,..., Aip} \; R = R' \quad \text{avec } R' (A_{i1}, A_{i2},..., A_{ip})$$

- **Jointure naturelle :** de 2 tables R et S est une table T telle que :
  - le schéma de T = union des schémas de R et de S
  - les tuples de T = concaténation des tuples de R et de S ayant même valeur pour attribut de même nom.

Soit: $R(A_1, A_2,..., A_n)$; $S(B_1, B_2,..., B_m)$;

$R \bowtie S = T ((A_1, A_2,..., A_n) \cup (B_1, B_2,..., B_m))$

- **ce sont des opérations inverses :**

$$\Pi_{A1, A2,..., Ap} \; T = R$$

$$\Pi_{B1, B2,..., Bm} \; T = S$$

Décomposition d'une table: décomposition 1

A	#véhicule	marque	type	couleur	nb_portes
	25AB75	Volvo	240GLE	bleu	4
	701VH13	Peugeot	605S	blanc	4
	703BT28	Peugeot	306C	vert	3
	34HT66	Renault	R19	blanc	5
	521BE13	Volvo	240GLE	bleu	4

$\Pi_{(marque, type, couleur)}$ A

B	marque	type	couleur
	Volvo	240GLE	bleu
	Peugeot	605S	blanc
	Peugeot	306C	vert
	Renault	R19	blanc

$\Pi_{(\#veh., couleur, nb\_portes)}$ A

C	#véhicule	couleur	nb_portes
	25AB75	bleu	4
	701VH13	blanc	4
	703BT28	vert	3
	34HT66	blanc	5
	521BE13	bleu	4

$A' = B \bowtie_{couleur} C \neq A$

A'	#véhicule	marque	type	couleur	nb_portes
	25AB75	Volvo	240GLE	*bleu*	*4*
	701VH13	Peugeot	605S	*blanc*	*4*
	701VH13	**Renault**	**R19**	***blanc***	***4***
	...	...	...	...	...
	..	...	...	...	...

→ **A' ≠ A : perte d'information !!!!**

Décomposition d'une table: décomposition 2

A	#véhicule	marque	type	couleur	nb_portes
	25AB75	Volvo	240GLE	bleu	4
	701VH13	Peugeot	605S	blanc	4
	703BT28	Peugeot	306C	vert	3
	34HT66	Renault	R19	blanc	5
	521BE13	Volvo	240GLE	bleu	4

$\Pi_{(type, marque, nb\_portes)}$ A

B	#type	marque	nb_portes
	240GLE	Volvo	4
	605S	Peugeot	4
	306C	Peugeot	3
	R19	Renault	5

$\Pi_{(\#veh., type, couleur)}$ A

C	#véhicule	type	couleur
	25AB75	240GLE	bleu
	701VH13	605S	blanc
	703BT28	306C	vert
	34HT66	R19	blanc
	521BE13	240GLE	bleu

$A' = B \bowtie_{type} C = A$

A'	#véhicule	marque	type	couleur	nb_portes
	25AB75	Volvo	240GLE	bleu	4
	701VH13	Peugeot	605S	blanc	4
	703BT28	Peugeot	306C	vert	3
	34HT66	Renault	R19	blanc	5
	521BE13	Volvo	240GLE	bleu	4

→ **A' = A : pas de perte d'information !!!!**

Décomposition d'une table (Ullman 80)

• remplacement d'une table R (A1, A2 … An) par une collection de tables R1, R2,… Rm

- Ri = $\prod_{\alpha}$ R
- R1 ⋈ R2 ⋈ … ⋈ Rm = R' même schéma que R

Décomposition sans perte d'une table

- décomposition R → R1, R2 … Rm /
  - R1⋈R2⋈ … ⋈Rm = R
  - ceci pour toute extension de R

pour cela:

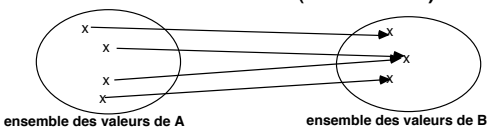
- besoin de **sémantique** sur les données:

-> notion de dépendance fonctionnelle

Dépendance fonctionnelle: DF (CODD 70)

Un attribut (ou groupe d'attributs) B d'une table R, est **fonctionnellement dépendant** d'un autre attribut (ou groupe d'attributs) A de R, **si à tout instant, chaque valeur de A n'a qu'une valeur associé de B :**

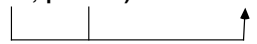
on note A → B **(A identifie B)**



Définition formelle : A → B <=> ∀ r ∀ t_a ∀ t_b (t_a{A} = t_b{A} => t_a{B} = t_b{B}) (avec r une réalisation de la table R):

#véhicule → type

(nom, prénom) → adresse
(nom, prenom) adresse



Remarque: une dépendance fonctionnelle « DF » est une **assertion intentionnelle** sur toutes les valeurs possibles des attributs et non pas sur les valeurs actuelles → impossible de réduire les DF d'une réalisation particulière d'une réalisation *(DF définie en intention pas en extension: on n'a pas DF→valeur)*

Dépendances fonctionnelles : DF (CODD 70)

- **Table EMPLOYE (#emp, nom, salaire, projet, date fin) :**

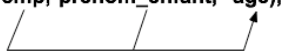
nom	***est dépendant de***	**#emp**
#emp	"	**nom**
salaire	"	**#emp**
projet	"	**#emp**
date fin	"	**#emp ou p**

Table EMPLOYE (# emp, nom, salaire, projet, date_fin),

- **Table ENFANT (#emp, prénom_enfant, âge) :**

âge ***est dépendant de*** **#emp et prenom_enfant (clé composée) :**

Table ENFANT (# emp, prenom_enfant, age),

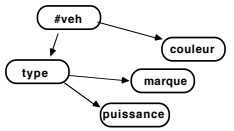


Graphes de DF

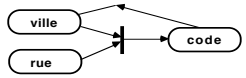
- **DF élémentaire** (Melkanoff 82) : X → A avec A ⊄ C et ∄ X' , X' ⊂ X / X' → A
- **Exemple de graphes de DF :**

D= {#veh → type,type → marque,type → puis,#veh →couleur)

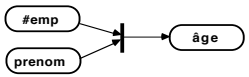
(D = {ens. de DF élémentaires})



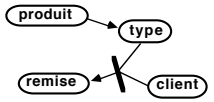
D = {(ville, rue) → code, code → ville} (code postal)



D = {(#emp, prénom_enfant) → âge)}



D = {produit → type, (type, client) → remise)}



Propriétés des DF

Axiomes d'Armstrong :

- **Réflexivité** :

$$A \rightarrow A$$

- **Augmentation** :

si $A \rightarrow B$ alors $(A,C) \rightarrow B,C$

- **Transitivité** :

si $A \rightarrow B$ et $B \rightarrow C$ alors $A \rightarrow C$

- **Pseudo-transitivité** :

si $A \rightarrow B$ et $(B,C) \rightarrow D$ alors $(A,C) \rightarrow D$

- **Union** :

si $A \rightarrow B$ et $A \rightarrow C$ alors $A \rightarrow B,C$

- **Décomposition** :

si $A \rightarrow B$ et $C \rightarrow B$ alors $A \rightarrow C$

Définition plus formelle d'une clé primaire

Soit R (A1, A2, A3 ...An) un schéma de la table avec l'ensemble D des DF , et X un sous-ensemble de A1, A2, A3 ...An, on dit que **X est une clé de R ssi** :

- **$X \rightarrow$ A1, A2, A3 ...An** et
- **il n'existe pas de sous-ensembles $Y \subset X$ / $Y \rightarrow$ A1, A2, A3 ...An**

$$X \subset \{A_1, A_2,..., A_n\} \ / \ X \rightarrow A_1, A_2,..., A_n \text{ et } \nexists\, Y \subset X \ / \ Y \rightarrow A_1, A_2,..., A_n$$

Soit :

Une clé est un **ensemble minimum d'attributs qui détermine tous les autres**

Ex : dans table VEHICULE :

(#véhicule) ... clé identifiant véhicule ?

(#véhicule, type) ... clé identifiant véhicule ?

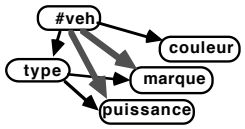
Il peut y avoir **plusieurs clés** pour une même table :

- on en choisit une comme "**clé primaire**"
- **en général** la clé primaire est la clé (plaçante) de la méthode d'accès du fichier implémentant la table (**clé primaire = clé plaçante**), mais pas toujours !!!
- on parle parfois de "**clé candidate**" pour les autres

Propriétés des dépendances fonctionnelles

Fermeture transitive : F^+ = {DF élémentaires} U {DF déduites par transitivité}

Ex :



D1, D2 ens. de DF élémentaires (D1 ≠ D2)

D1 ⇔ D2 s'ils ont même fermeture transitive

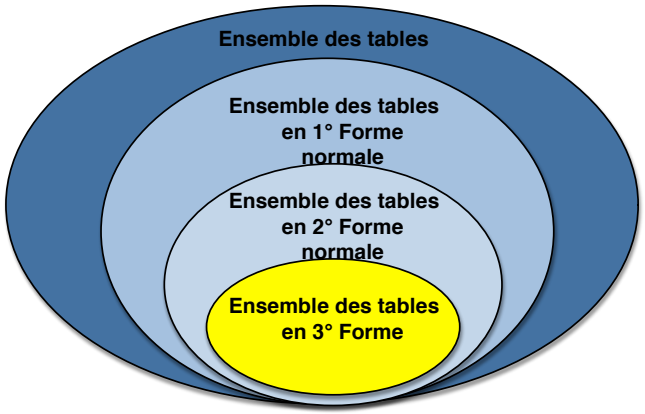
Couverture minimale :

D sous ensemble minimum de DF permettront de générer toutes les autres /
- aucune DF de D n'est redondante: $F \in D$; D - {F} non équivalent à D
- toute DF des attributs est dans la fermeture de D (D^+)
- en général pas unique
- permet la décomposition sans perte d'information

Ex :

F= {#veh → type,type → marque,type →puiss.,#veh → couleur}

Normalisation



- **Processus réversible**
- **Pas de perte d'information**

1NF- 1°forme normale

1NF = Élimination des groupes répétitifs : un attribut ne peut prendre que des valeurs atomiques

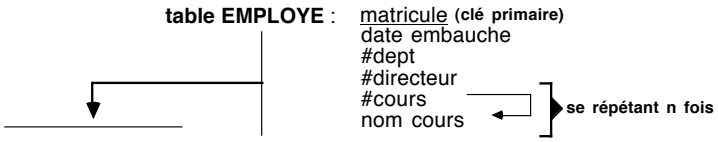


table EMPLOYE :
matricule **(clé primaire)**
date embauche
#dept
#directeur

table FORMATION :
matricule
#cours
nom cours
clé primaire composée

mais :

- *pour ajouter un nom cours, il faut attendre que quelqu'un le suive !…*
- *si on efface le seul tuple formation spécifiant un cours, on le perd !…*
- *mise à jour difficile du nom d'un cours suivi par plusieurs personnes !*

2NF- 2°forme normale

R 2NF⇔ R 1NF et chaque attribut extérieur à la clé primaire est **complètement dépendant** de la clé primaire

table EMPLOYE :
matricule **(clé primaire)**
date embauche
#dept
#directeur

table FORMATION :
matricule
#cours
nom cours

table FORMATION :
matricule
#cours

table COURS :
#cours
nom_cours

mais:

- *on ne peut pas rentrer un directeur tant qu'il n'a pas un employé sous sa responsabilité …*
- *si on efface le seul tuple employé relatif à un département on perd le directeur …*
- *si on veut changer le directeur d'un département, il faut le changer pour tous les tuples des employés de ce département …*

3NF- 3°forme normale

R 3NF⇔ R 2NF et aucun attribut extérieur à la clé primaire **ne dépend transitivement** de cette clé primaire

table EMPLOYE :
matricule
date embauche
#dept
#directeur

table EMPLOYE:
matricule
date embauche
#dept

table DEPARTEMENT :
#dept
#directeur

table FORMATION :
matricule
#cours

table COURS :
#cours
nom_cours

Synthèse normalisation

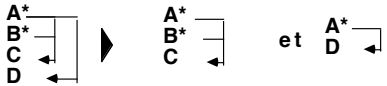
Tables non normalisées

↓

Eliminer les structures répétitives:
1NF

↓

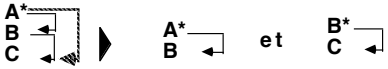
Éliminer les DF entre attributs non clé primaire et s'assurer de DF complètes entre attributs non clé primaire et clés primaires



2NF

↓

Éliminer les DF transitives entre les attributs non primaires et clés primaires



3NF

Forme normale de BOYCE-CODD

R BCNF⇔R 3NF et les seules DF élémentaires sont celles dans lesquelles une clé **détermine** un attribut

cru
pays
région

table VINS :

VINS	cru	pays	région
	corbière	france	languedoc
	broully	france	beaujolais
	chablis	usa	californie
	morgon	france	beaujolais
	chenas	france	beaujolais
	juliènas	france	beaujolais

cru
région

table CRUS :

CRUS	cru	région
	corbière	languedoc
	broully	beaujolais
	chablis	californie
	morgon	beaujolais
	chenas	beaujolais
	juliènas	beaujolais

région
pays

table REGION :

REGION	région	pays
	languedoc	france
	beaujolais	france
	californie	usa

Réduction de redondance : la dépendance (Cru, pays →région) est perdue, mais on peut la recomposer par jointure de CRUS et REGION sur l'attribut REGION.