CHAPITRE 5
Dépendances fonctionnelles
Décompositions
Traduction d'un diagramme de classe en schéma relationnel

Chapitre 5 : Plan

Introduction

L'objectif du chapitre est de se donner les outils permettant d'être capable de fournir un bon système de relations permettant de construire une base de données « cohérente » et « robuste ».
Il y a plusieurs techniques pour y arriver :

- Les formes normales

- Traduction d'un diagramme UML en un schéma relationnel

Nous allons dans ce chapitre exposer la deuxième méthode.
(la première sera exposée au second semestre)

Introduction

Un bon schéma relationnel doit :

- Éviter la redondance de l'information
- Ne pas perdre d'informations
- Être « robuste » : pouvoir insérer, modifier, supprimer des t-uples sans introduire d'incohérences dans les données.

Introduction

Concevoir une base de données c'est fournir une description des données cad des schémas ou diagrammes relationnels .

Les t-uples insérés ou modifiés doivent à tout instant vérifier toutes les contraintes exprimées par la description des données.

Introduction : exemple

Prenons par exemple la base livraison suivante :

Sachant que les frais de livraison sont fonction de la villeClient

Immat	NomAgence	marque	modèle	VilleClient	NomClient	FraisLivraison
237E98	Agence du pont vert	Jaguar	ZE	Grand Champ	Plouc	100
576T56	Agence Mike	Renault	Clio	Rennes	De Kerziziou	300
934Y76	Agence truc	Peugeot	407	Rennes	PasMoi	300
196F45	Agence Mike	Renault	Clio	Auray	PasToi	150

Il y a :

- Redondance d'information : le modèle clio associé à la marque Renault apparaît deux fois. De même pour les frais de livraison associés à rennes.
- Anomalies de modification : si les frais de livraison liés à Rennes changent il y a plusieurs t-uples à modifier. Si un client déménage, il faudra aller chercher les frais liés à cette ville.

Dépendances fonctionnelles(DF)

•Elles permettent d'exprimer des contraintes d'intégrité sur des attributs.

•Elles expriment ainsi certaines propriétés sur les données.

•Elles sont un cas particulier de contrainte d'intégrité

L'objectif de ce paragraphe est donc d'expliciter les DF de façon optimale, afin de trouver une décomposition de l'ensemble des attributs en tables sans perte d'information.

Dépendances fonctionnelles

Définition :

Soit R une relation et A l'ensemble des attributs de R, et soient X et Y deux sous ensembles de A.
On dit que Y dépend fonctionnellement de X ssi :

Étant donné une valeur de X, il lui correspond au plus une valeur de Y

On note $X \rightarrow Y$,
X est la **source**, Y est le **but** de la DF.

Dans l'exemple précédent : modèle $\rightarrow$ marque (et pas l'inverse !)

Dépendances fonctionnelles

Une dépendance fonctionnelle $X \rightarrow Y$ est dite **élémentaire** (dfe) ssi :

Pour tout sous ensemble X' strictement inclus dans X, il n'y a pas de DF entre X' et Y.

Par exemple la DF (immat,marque) $\rightarrow$ modèle n'est pas élémentaire car on a aussi la DF : immat $\rightarrow$ modèle.

On remarquera que toute DF ayant pour source un seul attribut est nécessairement élémentaire.

Dépendances fonctionnelles

DF et fonction mathématique :

Il y a une différence entre « dépendance fonctionnelle » entre deux attributs A1 et A2, et « fonction » : A1 |-----> A2

La df assure que, à un instant donné, la relation binaire entre A1 et A2 est une fonction.

Cependant , la valeur de A2 n'est pas fixée de façon absolue : par exemple, le client qui habitait Vannes hier, peut habiter Rennes aujourd'hui.

Dépendances fonctionnelles

DF intra table et DF inter table :

Les DF étant des dépendances entre les valeurs des attributs, il y a deux cas possibles :

- soit les valeurs des attributs concernent le même tuple (df **INTRA**)

- soit les valeurs des attributs concernent des tuples différents (df **INTER**)

Dépendances fonctionnelles

L'exemple typique de DF intra table est :

PK (tuple) ---> autre attribut du tuple

codage : contrainte **PK**

L'exemple typique de DF inter table est :

FK (tuple1) ---> attribut (tuple2)

avec FK (tuple1) = PK (tuple2)

codage : contrainte **FK**

Dépendances fonctionnelles

Exemple de DF inter table :

Vol(noVol (1), aeroportDep = @Aeroport[Nom])
Aeroport(Nom(1), capacité)

Il y a bien une DF inter table entre aeroportDep et Nom :

En effet à chaque attribut aéroportDep est associé au plus un nom, en revanche un Nom peut être associé à plusieurs aeroportDep.

Dépendances fonctionnelles

Une **DF forte** correspond à une fonction totale

codage : contrainte **NN** sur le but de la DF

Représentation : 

Une **DF faible** correspond à une fonction non totale

codage : aucun puisque aucune contrainte

Représentation : ---------->

Dépendances fonctionnelles

Exemples

Dans la relation :

Voitures(NoIm(1), marque (NN) , modèle (NN), NomPropriétaire)

Les DF intra tables :

NoIm ⟶ marque et NoIm⟶modèle sont FORTES

Mais la DF intra table : NoIm--------> NomPropriétaire est FAIBLE

Dépendances fonctionnelles

Une df INTRA : A -----> B

est **directe** s'il n'existe aucun attribut non clé (ou ensemble d'attributs non clé) C différent de B tel que :

A -----> C et **C -----> B**

Autrement dit la DF A---->B ne peut pas s'obtenir par transitivité

Dépendances fonctionnelles

Par exemple, Dans la relation :

Voitures(NoIm(1), marque (NN) , modèle (NN), NomPropriétaire)

La DF NoIm ⟶ marque n'est pas directe,

Car il y a aussi une DF modèle ⟶ marque !!

Par contre la DF NoIm ------> NomPropriétaire est bien directe

Dépendances fonctionnelles

Axiomes d'Armstrong
(dépendances fortes ou faibles)

la réfléxivité :

$$\text{Si : } B \subseteq A \text{ alors : } A \longrightarrow B$$

Augmentation :

$$\text{Si } A \longrightarrow B \text{ , alors } \forall Z \; AZ \longrightarrow BZ$$

Transitivité :

$$\text{Si } A \longrightarrow B \text{ et } B \longrightarrow C \text{ alors } A \longrightarrow C$$

Dépendances fonctionnelles

Axiomes d'Armstrong
(dépendances fortes ou faibles)

Pseudotransitivité :

Si $A \rightarrow B$ et $BC \rightarrow E$ alors : $AC \rightarrow E$

Décomposition :

Si $A \rightarrow B$ et $C \subseteq B$, alors $A \rightarrow C$

Union :

Si $A \rightarrow B$ et $A \rightarrow C$, alors $A \rightarrow BC$

Dépendances fonctionnelles

Par les axiomes d'Armstrong certaines dépendances fonctionnelles se déduisent d'autres.

On distingue donc les dépendances explicitées DF et les dépendances déduites DF+.

DF+ est la fermeture transitive de DF.

On dit que deux ensembles de DF, DF1 et DF2 sont équivalents ssi ils ont même fermeture transitive.

Dépendances fonctionnelles

L'objectif est d'obtenir une « couverture minimale » des DF nécessaires DF1, c'est à dire le plus petit ensemble DF' équivalent à DF1 , DF' vérifiera :

•Toutes les parties droites (but) de DF' sont réduites à un élément.

•Aucune partie gauche (source) de DF' ne contient d'élément redondant :

$\forall A \rightarrow B \in DF1$ et $A' \subset A$, $(DF' \setminus \{A \rightarrow B\} \cup \{A' \rightarrow B\})+ \neq DF'+$

•Il n'y a pas de dépendances superflues :

$\forall A \rightarrow B \in DF1$, $(DF1 \setminus \{A \rightarrow B\})+ \neq DF'+$

Dépendances fonctionnelles

Algorithme d'obtention d'une couverture minimale :

1. Pour enlever les parties droites à un élément on expanse toutes les DF A→B1,...,Bn en A→B1,...,A→Bn

2. Pour enlever les éléments redondants de gauche on essaye de les enlever en les prenant tour à tour

3. Pour enlever les dépendances superflues on cherche à en enlever tour à tour.

Le tout sans évidement modifier la fermeture transitive de l'ensembles de DF initial.

Dépendances fonctionnelles

Fermeture d'un ensemble d'attributs :

Soit une relation définie par A un ensemble d'attributs et DF un ensemble de dépendances fonctionnelles.
La fermeture B+ d'une partie B $\subset$ A relativement à DF1 $\subset$ DF est l'ensemble des attributs impliqués par DF1:

$$B+ = \{a \in A \text{ tq } B \rightarrow a \in DF1+\}$$

Exemple : A = {B,C,D,E} et DF1 = {B$\rightarrow$C,C$\rightarrow$D,C$\rightarrow$E}

{B}+ = {B,C,D,E}, {C}+ = {C,D,E}, {D}+ = {D} et {E}+ = {E}

Dépendances fonctionnelles

Algorithme d'obtention de B+ :

Données : R(A, DF) , $B \subset A$ et $DF1 \subset DF$.

B' = B
Répéter
    B_{tmp} = B'
    Pour chaque df $y \rightarrow z$ de DF1 **Faire**
        Si $y \in B$ **Alors** $B' = B' \cup \{z\}$
    FinPour
Jusqu'à B_{tmp} = B' ou B' = A

Dépendances fonctionnelles

Clé d'une relation :

Soit une relation définie par A l'ensemble des attributs et DF l'ensemble des DF la caractérisant. On dit que $K \subset A$ est une clé de la relation SSI :

$$K \rightarrow A \subset DF$$

C'est à dire qu'une clé détermine chaque t-uple de façon unique.

Dépendances fonctionnelles

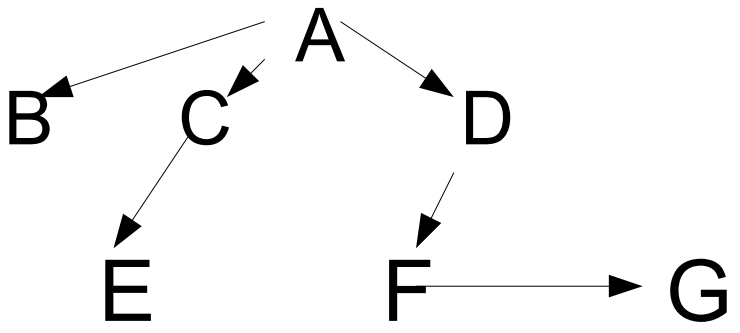
Graphe des dépendances fonctionnelles :

Pour chaque relation, on recense toutes les DF élémentaires qui ne se déduisent pas des autres.

On les représente sous forme d'un graphe orienté

Une relation peut avoir plusieurs graphes minimums : ils sont équivalent et représentent un ensemble de DF équivalentes.

Exemple :



Dépendances fonctionnelles

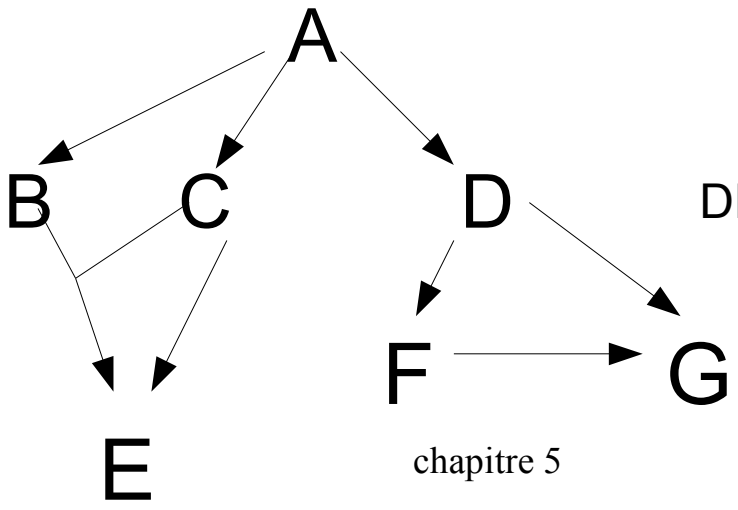
Graphe des dépendances fonctionnelles :

Il faut vérifier que le graphe est bien minimum.

Il permet de trouver les clés des relations

Il permet de tester si la décomposition est bonne, en en trouver une si nécessaire

Exemple de graphe non minimum :



DF déduite !!!

DF non élémentaire !!

Dépendances fonctionnelles

On obtiendra une <u>décomposition sans perte d'information</u> d'une relation R(A_1,...,A_n, DF) en R(U_1) et R(U_2) si :

$$U_1 \cup U2 = \{A_1,...,A_n\} \text{ et } R(U_1) * R(U_2) = R(A_1,...,A_n)$$

Ou bien de façon équivalente :

$$U_1 \cap U_2 \rightarrow U_1 - U_2 \in DF+ \text{ ou } U_1 \cap U_2 \rightarrow U_2 - U_1 \in DF+$$

Cela veut dire plus concrètement que l'on garde la possibilité de faire les mêmes requêtes avec la décomposition.
(via une jointure qui exprime un contrôle de cohérence).

Dépendances fonctionnelles

On obtiendra une <u>décomposition préservant les DF</u> si :

$$DF_1^+ \cup DF_2^+ = DF^+$$

Un choix de décomposition sans perte d'information et préservant les dépendances permet uniquement de limiter les contrôles de cohérence à faire lorsque l'on modifie les tables.

Dépendances fonctionnelles

Comment faire une décomposition valide ?

Tout schéma relationnel R(A,B,C) où A,B et C sont des ensembles d'attributs est décomposable en R_1 = R[A,B] et R_2 = R[A,C] s'il existe une dépendance fonctionnelle A → B.

On aura : R(A,B,C) = R_1(A,B)* R_2(A,C)

Les requêtes posées sur R et celles posées sur R_1* R_2 donneront le même résultat.

Dépendances fonctionnelles

Algorithme pour montrer la validité d'une décomposition :

Soit un ensemble d'attributs $\{A_1 ,...,A_n\}$ et une décomposition $D = \{U_1 ,...,U_n\}$.

	A1	.....	An
U1			
...			
Un			

Initialisation:

Tij = a_i si Ai $\in$ Ui , b_{ij} sinon.

Itération :

On cherche $X \longrightarrow Y \in DF$, et 2 lignes de T égales sur X afin de les égaliser sur les Y (on privilégie les a_i par rapport aux bij)

On boucle jusqu'à stabilisation de T, ou jusqu'à obtenir une ligne de a_i.

Dépendances fonctionnelles

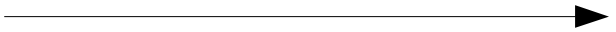
Exemple : Validité d'un décomposition

sur la relation R(F,V,C) avec DF = (F→V,V→C)
La décomposition {FV,VC} est-elle valide ?

	F	V	C
FV	a1	a2	b31
VC	b12	a2	a3

Il y a égalité sur la col V,

et V→ C ∈ DF, on égalise donc sur C



	F	V	C
FV	a1	a2	b31
VC	a1	a2	a3

La décomposition est donc valide

Dépendances fonctionnelles

Exemple : préservation des dépendances

sur la relation R(F,V,C) avec DF = (F→V,V→C)
La décomposition {FV,VC} préserve t-elle les dépendances ?

DF[{FV}] = {F→V}
DF[{VC}] = {V→C}

donc on a bien DF[{FV}]+ ∪ DF[{VC}]+ = DF+

Dépendances fonctionnelles

Ce choix de décomposition valide qui préserve les dépendances permet de limiter les contrôles de cohérence à effectuer lors des modifications sur les tables.

Ces deux critères ne sont cependant pas suffisants pour assurer que une telle décomposition évite des anomalies de mise à jour.

Par exemple : R(F,V,C,B) et DF= {F→V,V→C,B→F} décomposée en {F,V,C} et {B,F} est une décomposition valide et préserve les DF.

Diagramme UML → Schéma relationnel

On applique des « règles de traduction »
Diagramme de classe/Schéma relationnel.

Le schéma obtenu doit préciser :

Le domaine des attributs
Les clés primaires et secondaires
Les clés étrangères
Les contraintes d'unicité et de présence obligatoire
Les contraintes statiques et les autres.

Diagramme UML → Schéma relationnel

On ne s'intéresse dans les diapos suivantes qu'aux nouvelles contraintes liées aux cardinalités.

Bien sûr, toutes les contraintes issues de l'étape de modélisation devront ensuite être rajoutées.

Diagramme UML → Schéma relationnel

On adopte dans ce cours, un point de vue cohérent qui vous permet de pouvoir opérer ce passage *diagramme de classes* – *schéma relationnel*.

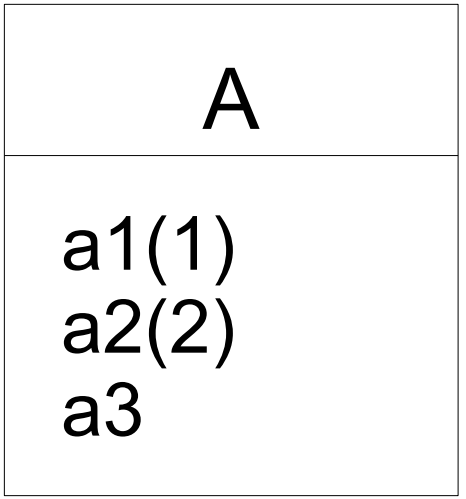
Il faut savoir qu'il y a d'autres implémentations possibles avec d'excellentes raisons. Nous en donnerons un exemple en remarque.

Diagramme UML → Schéma relationnel

Règle 1

Chaque classe est traduite par une relation. Les attributs de la classe deviennent des attributs de la relation, et les clés de la classe deviennent des clés de la relation.

Diagramme UML → Schéma relationnel



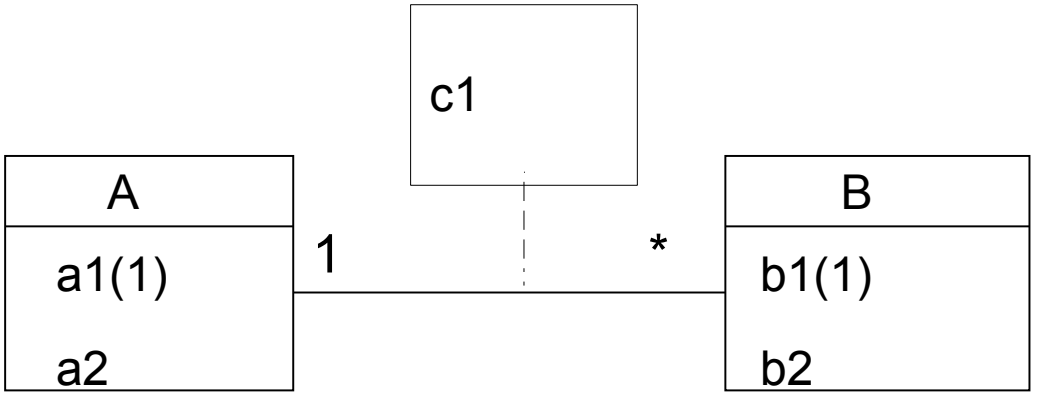
A(a1(1), a2(2), a3)

Diagramme UML → Schéma relationnel

Règle 2 : ASSOCIATION FONCTIONNELLE

Si une association est fonctionnelle alors elle se traduit par une clé étrangère dans la relation source.
Si des attributs portés apparaissent ils prennent place dans la relation source

Diagramme UML → Schéma relationnel



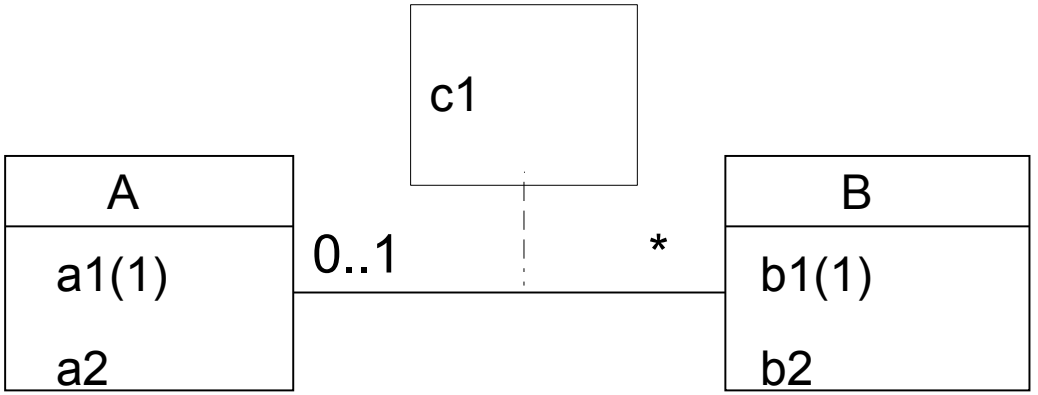
DF : RB → RA Forte Non surjective :

Implantation

RA(a1(1), a2)

RB(b1(1), b2, b3 = @A[a1] **NN**, c1)

Diagramme UML → Schéma relationnel



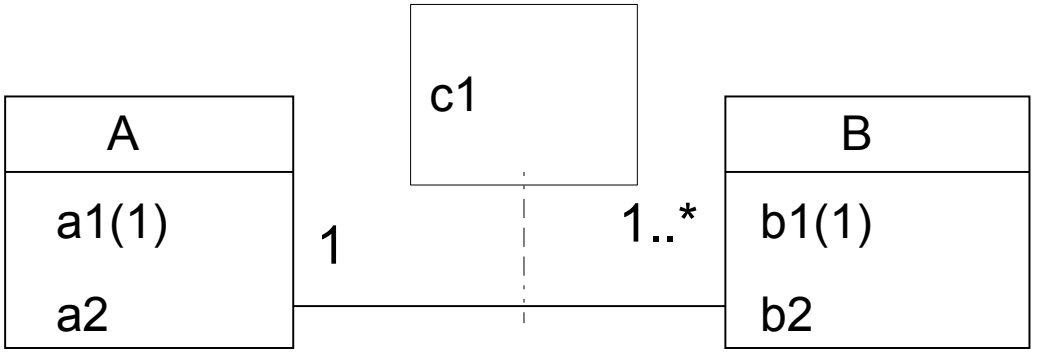
DF : RB → RA Faible Non surjective :

Implantation

RA(a1(1), a2)

RB(b1(1), b2, b3 = @A[a1], c1)

Diagramme UML → Schéma relationnel



DF : RB → RA Forte surjective :

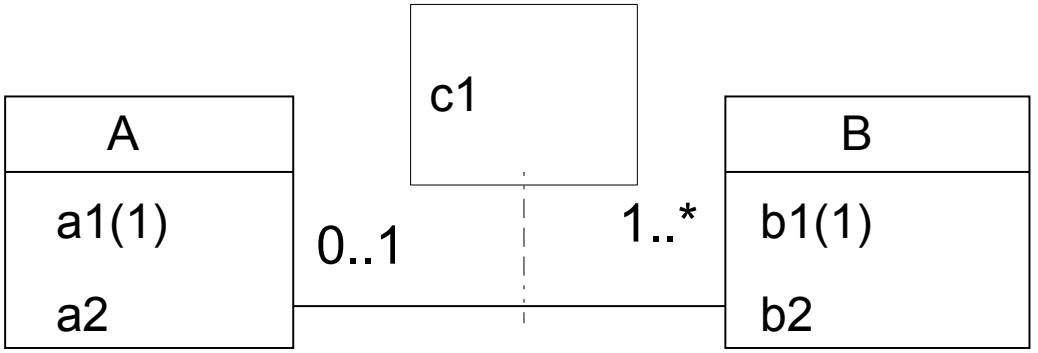
Implantation

RA(a1(1), a2)

RB(b1(1), b2, b3 = @A[a1] **NN**,c1)

avec la contrainte : A[a1] $\subseteq$ B[b3] (surjectivité)

Diagramme UML → Schéma relationnel



DF : RB → RA Faible surjective :

Implantation

RA(a1(1), a2)

RB(b1(1), b2, b3 = @A[a1],c1)

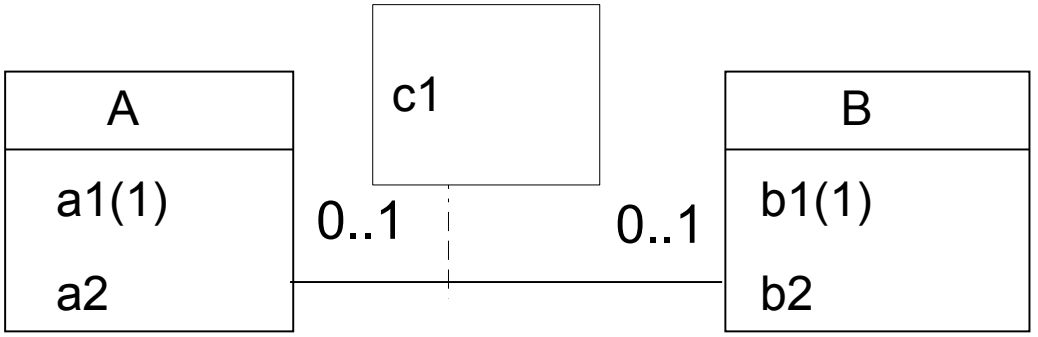
avec la contrainte : A[a1] $\subseteq$ B[b3] (surjectivité)

Diagramme UML → Schéma relationnel

Si l'association est doublement fonctionnelle, (multiplicité 0..1- 0..1), on a le choix de la table dans laquelle on va mettre la clé étrangère.

Le déport de clé se fait dans la table ou il y aura le moins de t-uples :

Diagramme UML → Schéma relationnel



association doublement fonctionnelle :

DF RB → RA Faible non surjective:

Implantation

RA(a1(1), a2)

RB(b1(1), b2, b3 = @A[a1] **UQ**, c1) avec

ou de façon équivalente :

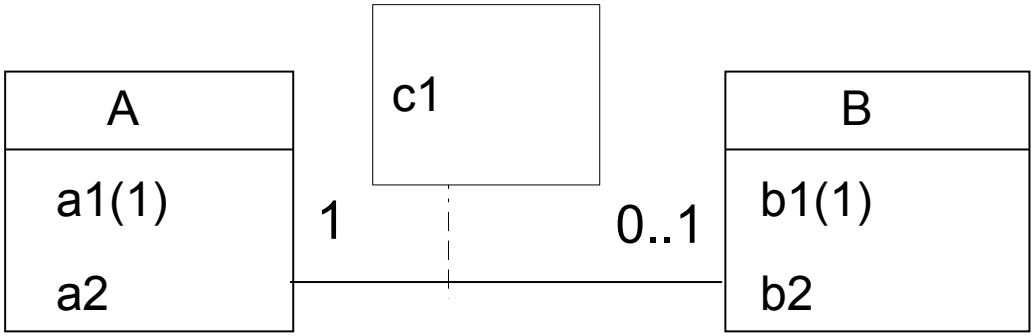
RA(a1(1), a2, a3 = @A[a1] **UQ**, c1)

RB(b1(1), b2)

Diagramme UML → Schéma relationnel

Si l'association est doublement fonctionnelle, (multiplicité 1- 0..1), il faut rajouter une contrainte d'unicité sur la clé étrangère : la clé étrangère devient une clé secondaire :

Diagramme UML → Schéma relationnel



association doublement fonctionnelle :

DF RB → RA Forte non surjective:

Implantation

RA(a1(1), a2)

RB(b1(1), b2, b3 = @A[a1] **NN UQ**, c1)

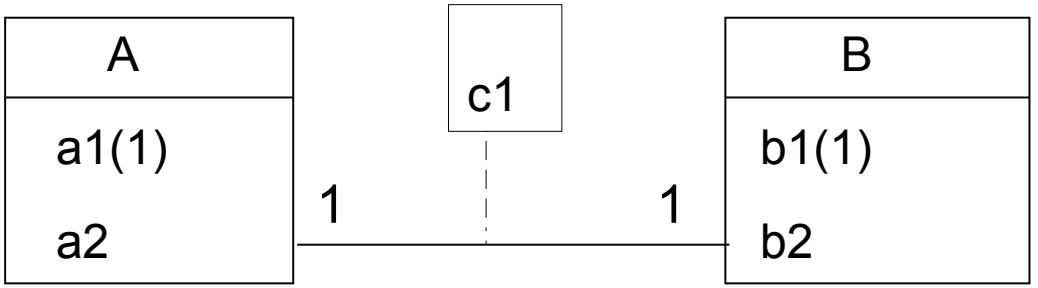
ou de façon équivalente : RA(a1(1), a2)

RB(b1(1), b2, b3 = @A[a1] (2),c1)

Diagramme UML → Schéma relationnel

Si l'association est doublement fortement fonctionnelle cad Forte et surjective donc bijective (multiplicité 1-1), on obtient une seule table :

Diagramme UML → Schéma relationnel



Implantation :

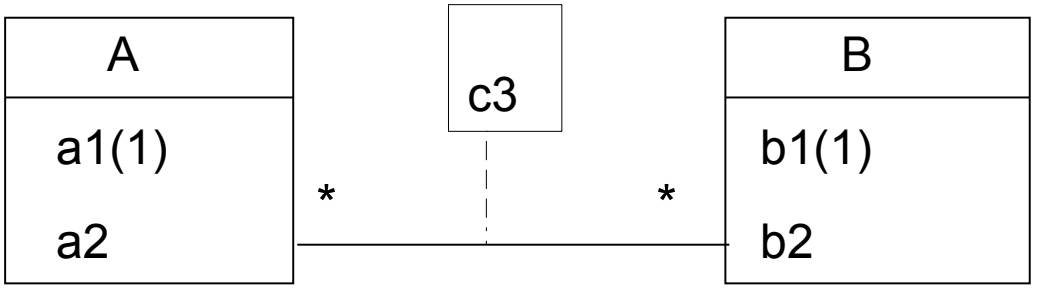
RA(a1(1),b1(2), a2, b2,c1)

Diagramme UML → Schéma relationnel

Règle 3 : ASSOCIATION NON FONCTIONNELLE

Si une association est non fonctionnelle, alors elle se traduit par la création d'une table association.
Si des attributs portés apparaissent ils prennent place dans cette nouvelle table.

Diagramme UML → Schéma relationnel



DF1 : RC → RA Forte non surjective

DF2 : RC → RB Forte non surjective

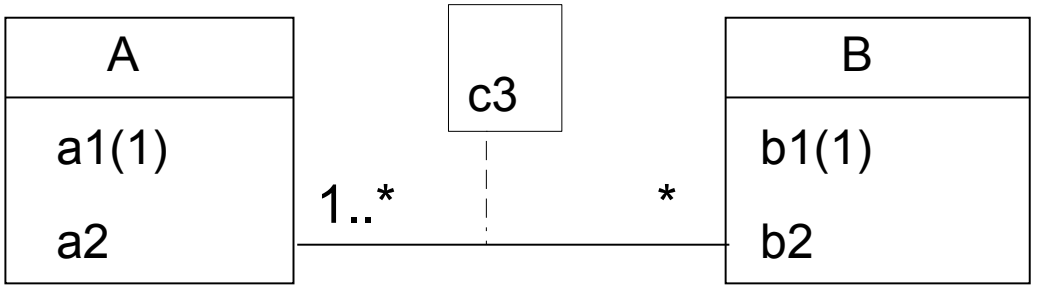
Implantation :

RA(a1(1),a2)

RB(b1(1), b2)

RC([c1 = RA.a1,c2 = RB.b1](1), c3)

Diagramme UML → Schéma relationnel



DF1 : RC → RA Forte non surjective

DF2 : RC → RB Forte surjective

Implantation :

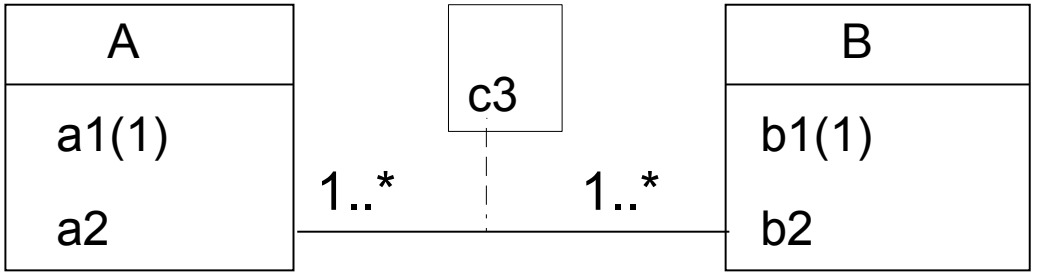
RA(a1(1),a2)

RB(b1(1), b2)

RC([c1 = RA.a1,c2 = RB.b1](1), c3)

avec RB[b1] $\subseteq$ RC[c2] (DF2 surjective)

Diagramme UML → Schéma relationnel



DF1 : RC → RA Forte surjective

DF2 : RC → RB Forte surjective

Implantation :

RA(a1(1),a2)

RB(b1(1), b2)

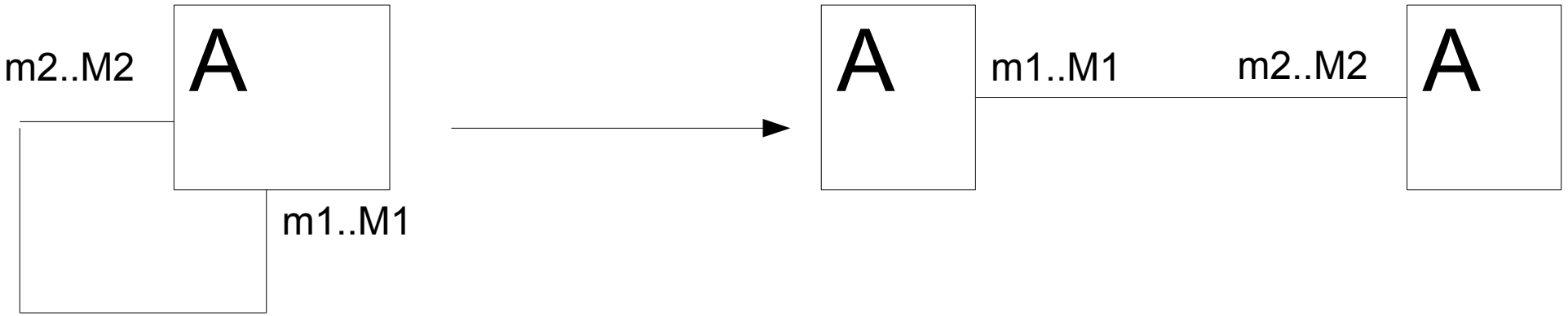
RC([c1 = RA.a1,c2 = RB.b1](1), c3)

avec RB[b1] $\subseteq$ RC[c2] et RA[a1] $\subseteq$ RC[c2]

Diagramme UML → Schéma relationnel

Règle 4 : ASSOCIATIONS INTERNES

Les associations internes correspondent au schéma suivant :



On commence par imaginer la même association en reliant deux occurrences de A

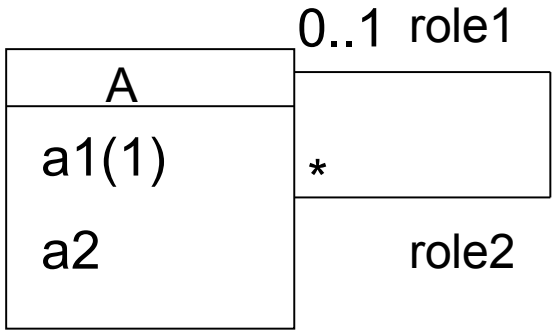
Diagramme UML → Schéma relationnel

Puis on traite ce cas comme les précédents, en partant du nouveau schéma.

Enfin, on réduit le schéma relationnel en supprimant les effets de duplication pour éviter les redondances.

Par exemple :

Diagramme UML → Schéma relationnel

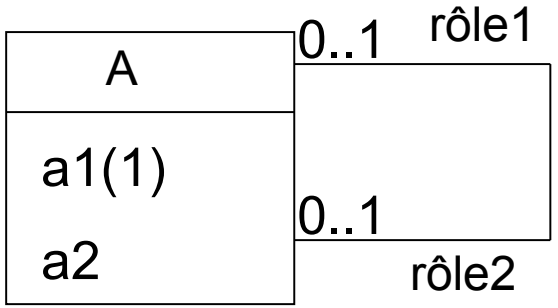


Implantation :

A(a1(1), a2, rôle1 = @A[a1])

etc ...

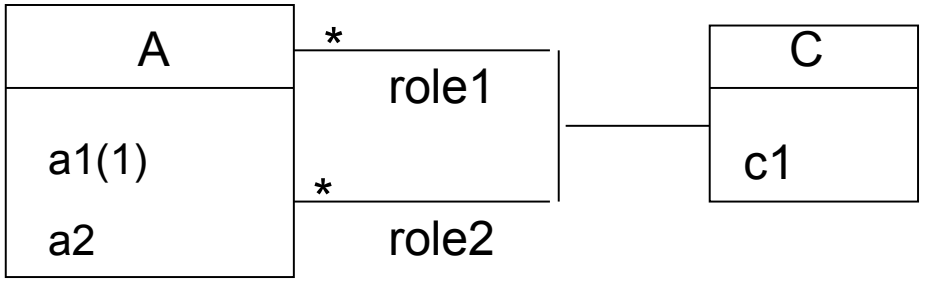
Diagramme UML → Schéma relationnel



Implantation :

A(<u>a1</u>, a2, rôle1 = @A[a1] **UQ**)

Diagramme UML → Schéma relationnel



Implantation :

A(a1(1), a2)

C(rôle1 = @A[a1] (1), rôle2 = @A[a2] (1), c1)

Diagramme UML → Schéma relationnel

La généralisation

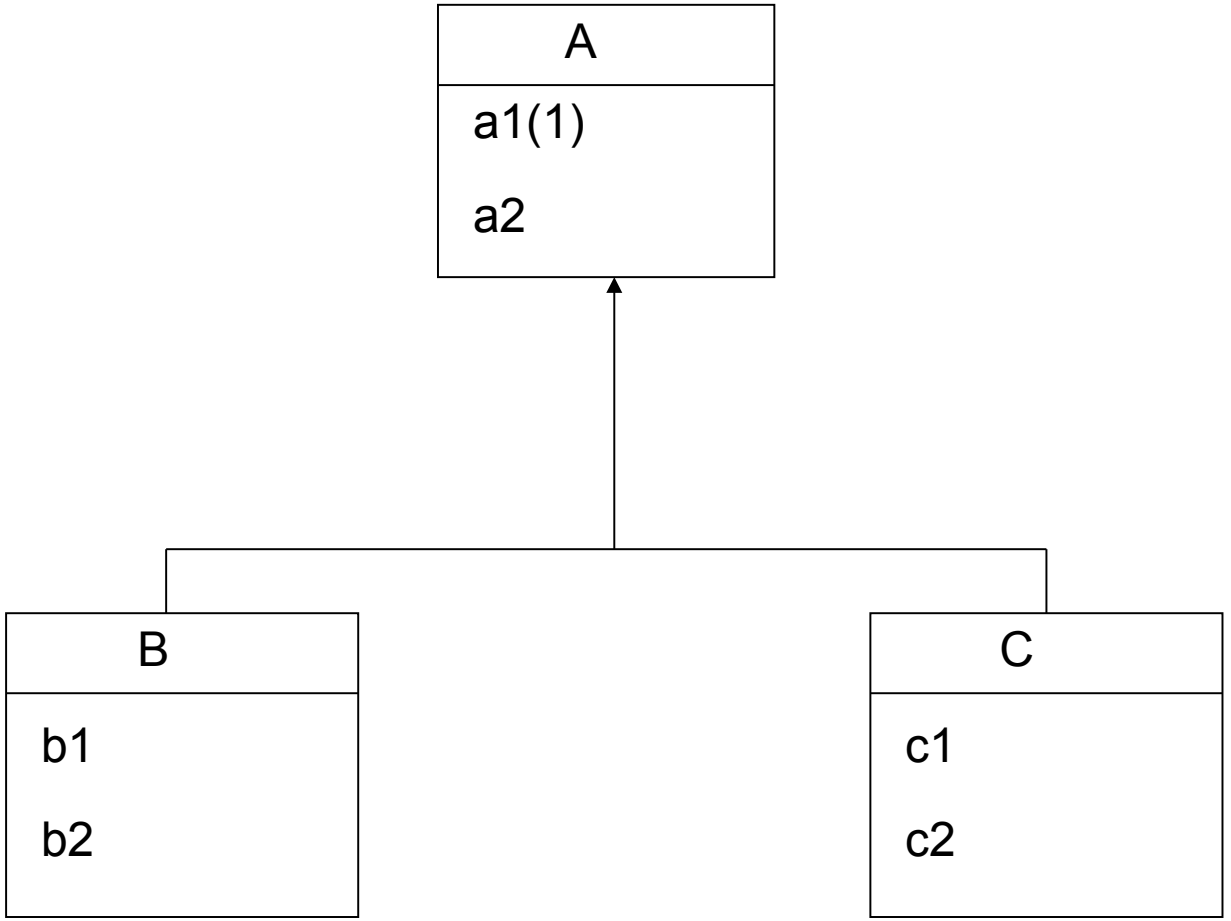


Diagramme UML → Schéma relationnel

Implantation :

A(<u>a1</u>, a2, type NN) $type \in \{B, C\}$

B(<u>@a1</u>, b1, b2)

C(<u>@a1</u>, c1, c2)

$$B[a1] \cap C[a1] = \varnothing$$

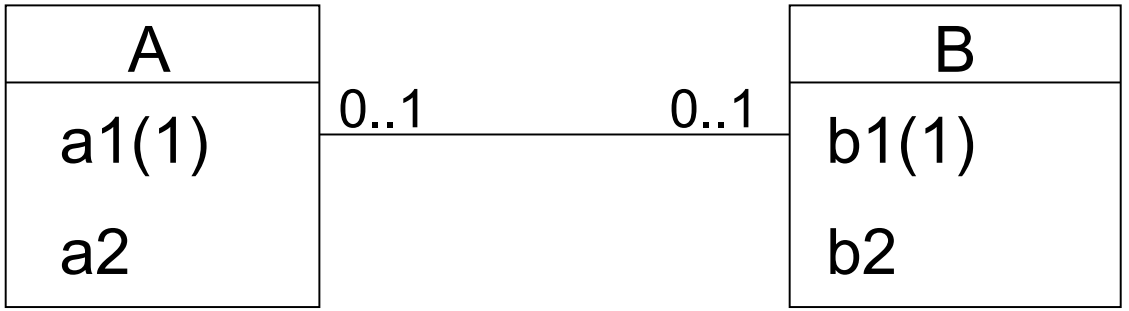
$$A[a1] \subset (B[a1] \cup C[a1]) \quad si \quad A \quad abstraite$$

Diagramme UML → Schéma relationnel

comme il a été dit précédemment, il y a d'autres possibilités suivant les cas.

Regardez le cas suivant et supposez, que la table A possède beaucoup de t-uples, et que l'association n'est réalisée que dans peu de cas. l'implémentation proposée utilisera moins de place mémoire, que pour celle proposée plus haut :

Diagramme UML → Schéma relationnel



Implantation :

A(a1(1), a2)

B(b1(1), b2)

C(c1 = @A[a1] (1) UQ, c2 = @B[B1] (1) UQ)

au lieu de

A(a1(1), a2, @b1 **UQ**)

B(b1(1), b2)

Diagramme UML → Schéma relationnel

EXERCICE :