

# CARDINALITES

Comprendre les relations et les  
cardinalités



# Exercices #3

Dans ces exercices, il vous est demandé de stocker les relations de tables d'un système dans leur globalité

Vous gérez le système de gestion de Carrefour.  
On vous demande de faire l'évolution suivante : il faut stocker les fournisseurs, les produits qu'ils apportent et les rayons dans lesquels il faut mettre les produits.

Comment stocker toutes ces informations ? (20 min)



Il s'agit tout d'abord d'identifier les entités/tables de la demande :

- Fournisseur
- Produit
- Rayon

Regardons les relations possibles en chaque pair de tables :

1. Fournisseur – Produit ?
2. Fournisseur – Rayon ?
3. Produit – Rayon ?

Regardons les relations possibles en chaque pair de tables :

1. Fournisseur – Produit ? (fournir)
2. Fournisseur – Rayon ? (aucun)
3. Produit – Rayon ? (composer)



Regardons les relations possibles en chaque pair de tables :

1. Fournisseur – Produit ? (fournir)
- ~~2. Fournisseur – Rayon ? (aucun)~~
3. Produit – Rayon ? (composer)

Nous avons donc 2 relations que l'on souhaite stocker :

1. Fournisseur – Produit ? (fournir)
2. Produit – Rayon ? (composer)

On va traiter la 1<sup>er</sup> relation :

- Fournisseur – Produit ? (fournir)

Par exemple, une bonne réponse peut être :

- Un « Fournisseur » peut « fournir » combien de « Produit » ? Plusieurs produits
- Une « Produit » peut être « fourni » de combien de « Fournisseur » ? Plusieurs fournisseurs

C'est donc une relation ManyToMany (ou n-n) :

- Plusieurs fournisseurs
- Plusieurs produits

Pour une relation ManyToMany, nous allons créer une nouvelle structure.

# On stocke la relation « fournir »

id	...
ref	...

fournisseur

fournisseur_id	...
produit_id	...

Fournir\_fournisseur\_produit

id	...
titre	...
prix	...

produit

On va traiter la 2<sup>eme</sup> relation :

- Produit – Rayon ? (composer)

Par exemple, une bonne réponse peut être :

- Un « Produit » peut « composer/être dans » combien de « Rayon » ? 1 seul rayon
- Une « Rayon » peut être « composé » de combien de « Produit » ? Plusieurs produits



C'est donc une relation OneToMany (ou 1-n) :

- Plusieurs produits
- Un seul rayon

# On stocke la relation « composer »

id	...
ref	...

fournisseur

fournisseur_id	...
produit_id	...

Fournir\_fournisseur\_produit

id	...
nom	...

rayon

id	...
titre	...
prix	...
rayon_id	...

produit



Vous faites un site de location de voitures. Le client vous demande de gérer les voitures, les personnes qui s'inscrivent sur le site et leur réservation.

Comment stocker toutes ces informations ? (20 min)



Il s'agit tout d'abord d'identifier les entités/tables de la demande :

- Voiture
- Utilisateur
- Reservation

Regardons les relations possibles en chaque pair de tables :

1. Voiture – Utilisateur ?
2. Voiture – Reservation ?
3. Utilisateur – Reservation ?

Regardons les relations possibles en chaque pair de tables :

1. Voiture – Utilisateur ? (aucun lien de manière générale, sans le cas de la réservation)
2. Voiture – Reservation ? (reserver)
3. Utilisateur – Reservation ? (effectuer)

Regardons les relations possibles en chaque pair de tables :

- ~~1. Voiture – Utilisateur ? (aucun lien de manière générale, sans le cas de la réservation)~~
2. Voiture – Reservation ? (reserver)
3. Utilisateur – Reservation ? (effectuer)

Nous avons donc 2 relations que l'on souhaite stocker :

1. Voiture – Reservation ? (reserver)
2. Utilisateur – Reservation ? (effectuer)



On va traiter la 1<sup>er</sup> relation :

- Voiture – Reservation ? (reserver)

Par exemple, une bonne réponse peut être :

- Un « Reservation » peut « réserver » combien de « Voiture » ? 1 seule voiture
- Une « Voiture » peut être « réservé » de combien de « Reservation » ? Plusieurs reservations (sur des jours différents par exemple)

C'est donc une relation OneToMany (ou 1-n) :

- Plusieurs reservations
- Une seule voiture

# On stocke la relation « reserver »

id	...
modele	...
couleur	..
puissance	..

voiture

id	...
date	...
prix	...
voiture_id	...

reservation

- On va traiter la 2<sup>eme</sup> relation :
- Utilisateur – Reservation ? (effectuer)

Par exemple, une bonne réponse peut être :

- Un « Utilisateur » peut « effectuer » combien de « Reservation » ? Plusieurs réservation
- Une « Reservation » peut être « effectué » par combien de « Utilisateur » ? 1 seul utilisateur

C'est donc une relation OneToMany (ou 1-n) :

- Plusieurs reservations
- Un seul utilisateur

# On stocke la relation « effectuer »

id	...
modele	...
couleur	..
puissance	..

voiture

id	...
nom	...
email	..

utilisateur

id	...
date	...
prix	...
voiture_id	...
utilisateur_id	

reservation



Un client vous demande de réaliser un site pour gérer des recettes de cuisine. On vous demande de gérer les recettes et leurs étapes ainsi que les ingrédients.

Comment stocker toutes ces informations ? (20 min)



Il s'agit tout d'abord d'identifier les entités/tables de la demande :

- Recette
- Etape
- Ingredient

Regardons les relations possibles en chaque pair de tables :

1. Recette – Etape ?
2. Recette – Ingredient ?
3. Etape – Ingredient ?

Regardons les relations possibles en chaque pair de tables :

1. Recette – Etape ? (composer)
2. Recette – Ingredient ? (utiliser)
3. Etape – Ingredient ? (utiliser)

On va traiter la 1<sup>er</sup> relation :  
Recette – Etape ? (composer)

Par exemple, une bonne réponse peut être :

- Un « Etape » peut « composer » combien de « Recette » ? Plusieurs recettes
- Une « Recette » peut être « composé » de combien de « Etape » ? Plusieurs étapes

C'est donc une relation ManyToMany (ou n-n) :

- Plusieurs recettes
- Plusieurs etapes

Pour une relation ManyToMany, nous allons créer une nouvelle structure.

# On stocke la relation « composer »

id	...
titre	...
duree	...

recette

recette_id	...
etape_id	...

composition\_recette\_etape

id	...
contenu	...

etape



On va traiter la 2<sup>eme</sup> relation :  
Recette – Ingredient ? (utiliser)

Par exemple, une bonne réponse peut être :

- Une « Recette » peut « utiliser » combien de « Ingredient » ? Plusieurs ingrédients
- Un « Ingredient » peut être « utilisé » de combien de « Recette » ? Plusieurs recettes

C'est donc une relation ManyToMany (ou n-n) :

- Plusieurs recettes
- Plusieurs ingrédients

Pour une relation ManyToMany, nous allons créer une nouvelle structure.

# On stocke la relation « utiliser »

id	...
titre	...
duree	...

recette

recette_id	...
etape_id	...

composition\_recette\_etape

recette_id	...
ingredient_id	...

utilisation\_recette\_ingredient

id	...
contenu	...

etape

id	...
nom	...

ingredient

On va traiter la 3<sup>eme</sup> relation :  
Etape – Ingredient ? (utiliser)

Par exemple, une bonne réponse peut être :

- Une « Etape » peut « utiliser » combien de « Ingredient » ? Plusieurs ingrédients
- Un « Ingredient » peut être « utilisé » de combien de « Etape » ? Plusieurs étapes

C'est donc une relation ManyToMany (ou n-n) :

- Plusieurs étapes
- Plusieurs ingrédients

Pour une relation ManyToMany, nous allons créer une nouvelle structure.

# On stocke la relation « utiliser »

id	...
titre	...
duree	...

recette

recette_id	...
etape_id	...

composition\_recette\_etape

ingredient_id	...
etape_id	...

utilisation\_etape\_ingredient

id	...
contenu	...

etape

id	...
nom	...

ingredient

utilisattion\_recette\_ingredient



On vous demande d'intervenir sur un site existant de formation en ligne. On vous demande de gérer les videos de chaque cours, ainsi que le langage auquel elles se rapportent.

Comment stocker toutes ces informations ? (20 min)



Il s'agit tout d'abord d'identifier les entités/tables de la demande :

- Video
- Cours
- Langage

Regardons les relations possibles en chaque pair de tables :

1. Video – Cours ?
2. Video – Langage ?
3. Cours – Langage ?

Regardons les relations possibles en chaque pair de tables :

1. Video – Cours ? (composer)
2. Video – Langage ? (aucun, une video n'as pas de lien direct avec le langage, c'est plutôt le cours qui a un lien avec le langage)
3. Cours – Langage ? (introduire)

Regardons les relations possibles en chaque pair de tables :

1. Video – Cours ? (composer)
2. ~~Video – Langage ? (aucun, une video n'as pas de lien direct avec le langage, c'est plutôt le cours qui a un lien avec le langage)~~
3. Cours – Langage ? (introduire)

Nous avons ici 2 relations :

1. Video – Cours ? (composer)
2. Cours – Langage ? (introduire)

On va traiter la 1<sup>er</sup> relation :  
Video – Cours ? (composer)

Par exemple, une bonne réponse peut être :

- Une « Video » peut « composer » combien de « Cours » ? 1 seul cours
- Un « Cours » peut être « composé » de combien de « Video » ? Plusieurs vidéos



C'est donc une relation OneToMany (ou 1-n) :

- Plusieurs videos
- Un seul cours

# On stocke la relation « composer »

id	...
intitulé	...
couleur	..
puissance	..

cours

id	...
date	...
prix	...
cours_id	...

video

On va traiter la 2<sup>eme</sup> relation :  
Cours – Langage ? (introduire)

Par exemple, une bonne réponse peut être :

- Une « Cours » peut « introduire » combien de « Langage » ? 1 seul langage
- Un « Langage » peut être « introduit » de combien de « Cours » ? Plusieurs cours

C'est donc une relation OneToMany (ou 1-n) :

- Plusieurs cours
- Un seul langage

# On stocke la relation « composer »

id	...
intitulé	...
couleur	...
puissance	...
langage_id	...

cours

id	...
nom	...

langage

id	...
date	...
prix	...
cours_id	...

video

Vous souhaitez faire un reseau social comme facebook. Le client vous demande de gérer les personnes qui s'inscrivent sur le site et les demandes d'amis.

Comment stocker toutes ces informations ? (20 min)  
Plus dur



Il s'agit tout d'abord d'identifier les entités/tables de la demande :

- Utilisateur
- Demande (demande d'ami)

Les amis ne sont pas une entité à part, car les amis sont en fait des utilisateurs !



Regardons les relations possibles en chaque pair de tables :

1. Utilisateur – Demande ? (envoyer)

Regardons les relations possibles en chaque pair de tables :

1. Utilisateur – Demande ? (envoyer)
2. Utilisateur – Demande ? (recevoir)

Il s'agit bien de 2 relations différentes, mais entre 2 mêmes entités !

On va traiter la 1<sup>er</sup> relation :  
Utilisateur – Demande ? (envoyer)

Par exemple, une bonne réponse peut être :

- Un « Utilisateur » peut «envoyer » combien de « Demande » ? Plusieurs demande
- Un « Demande » peut être «envoyé » par combien de « Utilisateur » ? 1 seul utilisateur

