

Base de données

Chapitre 6 : Modélisation avancée

Joel Cavat

2022

Introduction

- Objectifs
 - Concepts de modélisation avancés
 - Généralisation/Spécialisation
 - Association ternaires (et n-aires)
 - Dépendance d'inclusion

Rappel

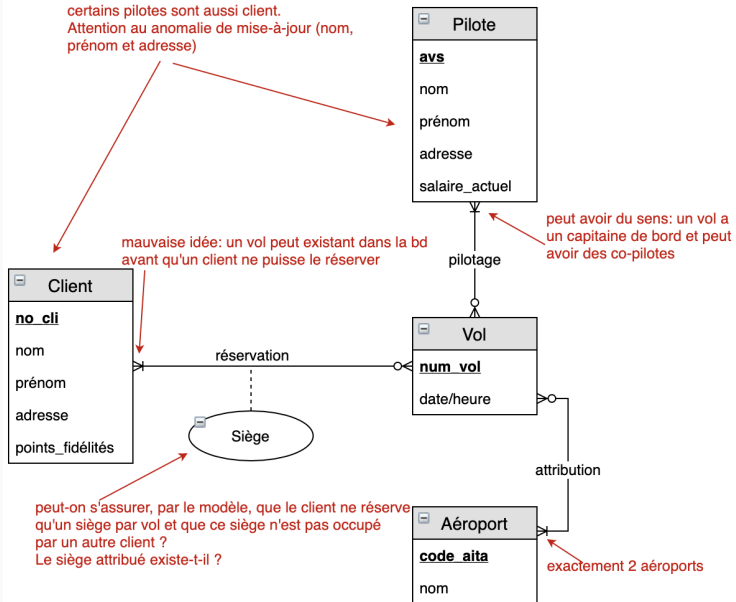
L'importance de la clé et de sa signification

1. Vente(article, fournisseur, prix, ...)
2. Vente(article, fournisseur, prix, ...)
3. Vente(article, fournisseur, prix, ...)
4. Vente(article, fournisseur, prix, ...)

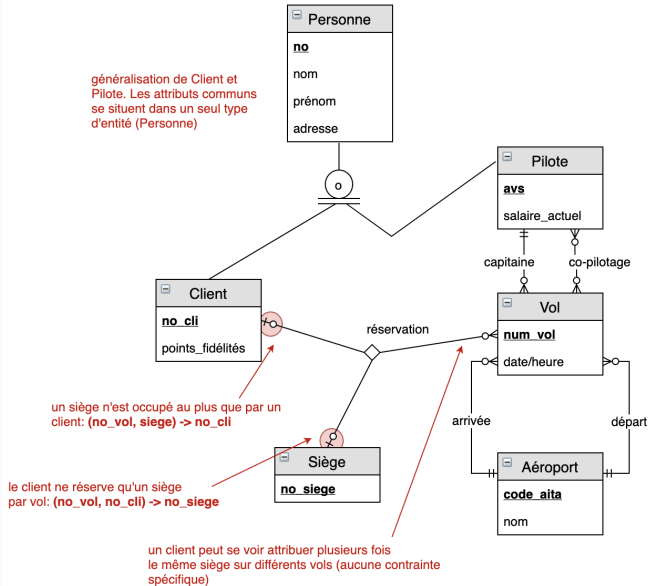
L'importance de la clé et de sa signification

1. Un article est vendu par un fournisseur à un prix donné
 - correspond à la DF: $\{\text{article}\} \rightarrow \{\text{fournisseur, prix, ...}\}$
2. Un article est vendu par plusieurs fournisseurs. Son prix dépend du fournisseur
 - correspond à la DF: $\{\text{article, fournisseur}\} \rightarrow \{\text{prix, ...}\}$
3. Un article peut être vendu à des prix différents par un même fournisseur
 - correspond à la DF: $\{\text{article, fournisseur, prix}\} \rightarrow \{\dots\}$
4. Cette relation comporte deux clés.
 - Un article est vendu par un fournisseur à un prix donné ; chaque article doit avoir un fournisseur différent
 - Un fournisseur ne peut vendre qu'un article

Cas introductif



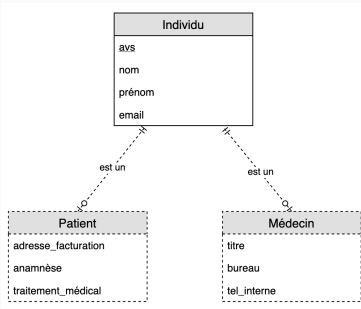
Cas introductif



Généralisation / Spécialisation

Principe d'entité générale vs entités spécifiques (\approx héritage)

- Associations de type "est un"
- Représentation possible (mais trop simpliste)



Ne renseigne pas :

- si un patient peut être également médecin
- si un individu peut être enregistré sans être ni médecin ni patient

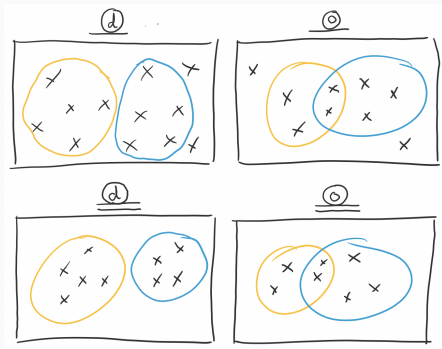
Amélioration du modèle

Donner une indication

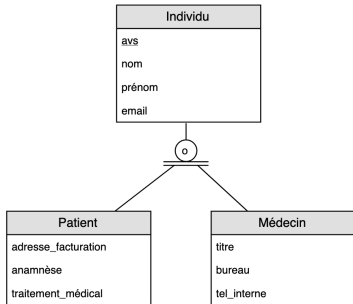
- sur les contraintes de **chevauchements** des entités-filles
 - *overlap*: chevauchement possible ; un médecin peut également être un patient
 - *disjoint*: chevauchement interdit
- sur la **participation** de l'entité parente
 - *partial*: enregistrement d'individus sans qu'ils soient patients ou médecins
 - *total*: impossibilité d'enregistrer des individus

Généralisation/Spécialisation

- ⊖ disjoint - partiel
- ⊗ overlap - partiel
- ⊖ disjoint - total
- ⊗ overlap - total

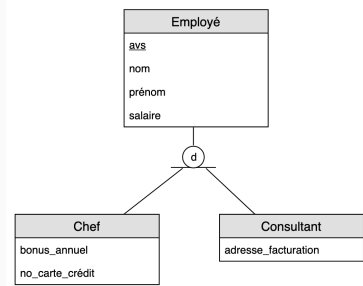


Exemple 1



- un individu est obligatoirement un patient et/ou un médecin.

Exemple 2



- un employé peut être un chef ou consultant
- possibilité d'enregistrer des employés qui ne sont ni chef ni consultant

Schéma relationnel associé: plusieurs solutions possibles 😬

Modèle EA

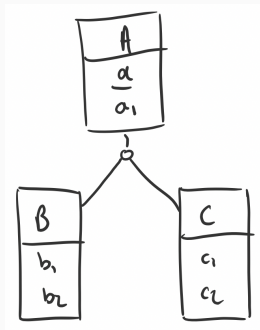


Schéma relationnel associé (version 1)

$A(\underline{a}, a_1)$

$B(\underline{a}, b_1, b_2), a \subseteq A.a$

$C(\underline{a}, c_1, c_2), a \subseteq A.a$

Remarques

- **overlap partiel**
- aucun attribut vide
- nécessite une jointure
- solution à privilégier

Modèle EA

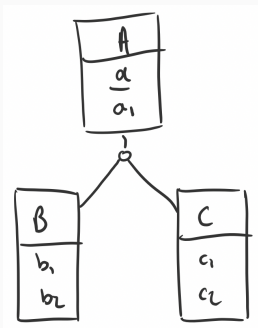


Schéma relationnel associé (version 2)

$A(\underline{a})$

$B(\underline{a}, a_1, b_1, b_2), a \subseteq A.a$

$C(\underline{a}, a_1, c_1, c_2), a \subseteq A.a$

Remarques

- **overlap total**
- aucun attribut vide
- jointure plus nécessaire
- gestion centralisée de la clé
- **redondance des attributs communs**

Modèle EA

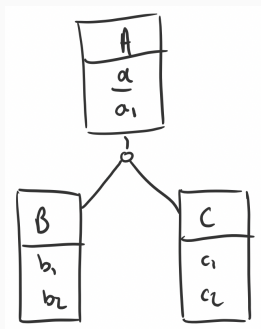


Schéma relationnel associé (version 3)

$B(\underline{a}, a_1, b_1, b_2)$

$C(\underline{a}, a_1, c_1, c_2)$

Remarques

- **overlap total**
- aucun attribut vide
- efficace
- le modèle EA aurait également pu éliminer A
- **redondance des attributs communs**

Modèle EA

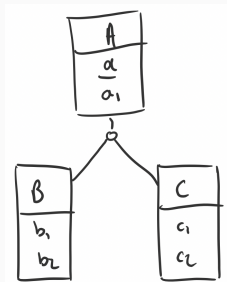


Schéma relationnel associé (version 4)

– disjoint total:

$A1(\underline{a}, a_1, b_1, b_2, c_1, c_2, \text{Enum}(B,C))$

– ou disjoint partiel:

$A2(\underline{a}, a_1, b_1, b_2, c_1, c_2, \text{Enum}(A,B,C))$

Remarques

- utilisation d'un discriminant
- **seule solution permettant d'être disjoint**
- disjoint total si $\text{Enum}(B,C)$
- disjoint partiel si $\text{Enum}(A,B,C)$
- Enum ne peut avoir qu'une valeur possible
- attributs vides ; risque d'incohérences
- non-standard SQL

Modèle EA

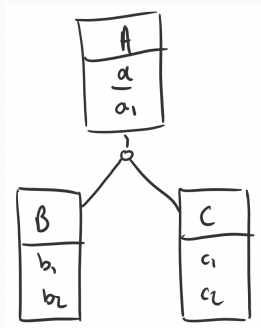


Schéma relationnel associé (version 5)

- overlap total $A1(\underline{a}, a_1, b_1, b_2, c_1, c_2, \text{Set}(B,C))$
- ou overlap partiel
 $A2(\underline{a}, a_1, b_1, b_2, c_1, c_2, \text{Set}(A,B,C))$

Remarques

- overlap total si $\text{Set}(B,C)$
- overlap partial si $\text{Set}(A,B,C)$
- Set peut contenir plusieurs valeurs
- attributs vides ; risque d'incohérences
- non-standard SQL

Associations récursives

Associations récursives

Application des règles standards (ci-dessous, exemple one-to-many)

Modèle EA - One-to-Many

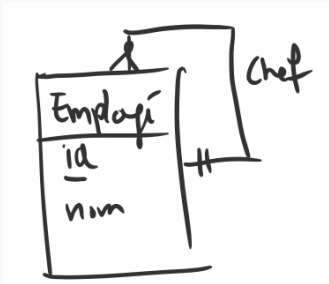


Schéma relationnel associé

Employe(id, nom, id_chef),
id_chef \subseteq Employe.id
id_chef optionel (un chef n'a pas forcément un chef)

ou

Employe(id, nom),
Cheffitude(id_emp, id_chef),
id_emp \subseteq Employe.id
id_chef \subseteq Employe.id

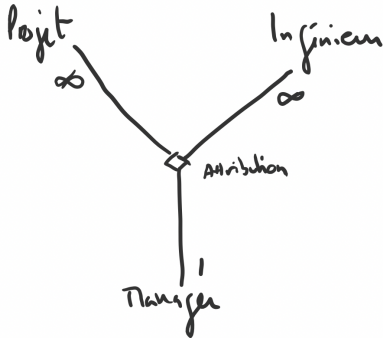
Remarques

- aucune transformation ne garantit qu'un employé a obligatoirement un chef

Associations ternaires (ou n-aires avec $n \geq 3$)

- Associations regroupant plus de deux TE
- Difficultés à déterminer les cardinalités
- Les cardinalités du côté d'un TE se déterminent à l'aide des autres TE

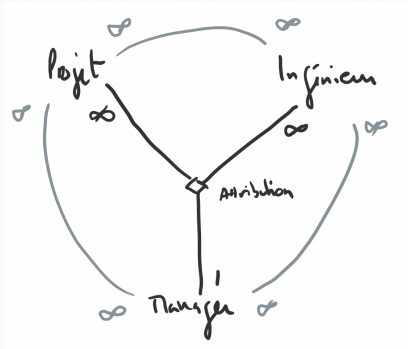
Modèle EA Many-Many-One



Interprétation

- Un ingénieur n'aura qu'un seul manager par projet
 - Connaissant un projet et un ingénieur, il n'existe qu'un manager attribué
 - Un manager sur un projet peut gérer plusieurs ingénieurs
 - Connaissant un ingénieur et un manager, ils peuvent se voir attribuer plusieurs projets

Modèle EA Many-Many-One



Cela ne signifie pas qu'un projet n'a qu'un manager

Interprétation

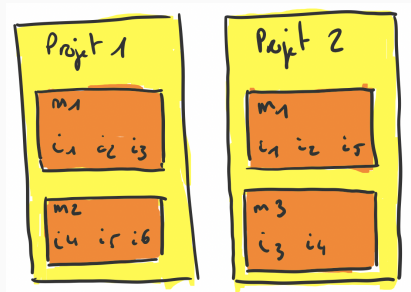
- Un ingénieur n'aura qu'un seul manager par projet
 - Connaissant un projet et un ingénieur, il n'existe qu'un manager attribué
- Un manager sur un projet peut gérer plusieurs ingénieurs
- Connaissant un ingénieur et un manager, ils peuvent se voir attribuer plusieurs projets

Associations ternaires

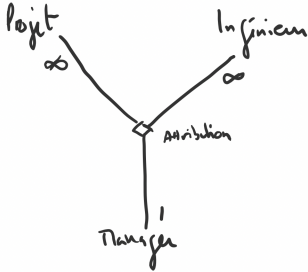
Modèle EA Many-Many-One



Interprétation



Modèle EA Many-Many-One



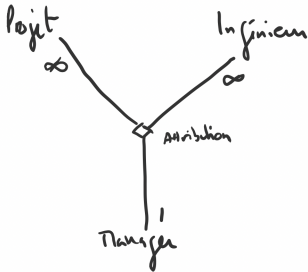
Cette proposition est-elle correcte ?

Attribution(id_projet, id_ingenieur, id_manager)

$\text{id_projet} \subseteq \text{Projet.id},$
 $\text{id_ingenieur} \subseteq \text{Ingenieur.id},$
 $\text{id_manager} \subseteq \text{Manager.id}$

Associations ternaires

Modèle EA Many-Many-One



Cette proposition est-elle correcte ?

Attribution(id_projet, id_ingenieur, id_manager)

$\text{id_projet} \subseteq \text{Projet.id}$,

$\text{id_ingenieur} \subseteq \text{Ingénieur.id}$,

$\text{id_manager} \subseteq \text{Manager.id}$

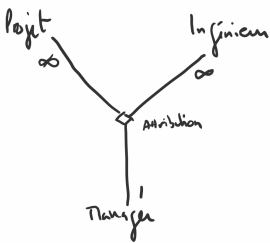
Elle est fausse

id_projet	id_ingenieur	id_manager
p1	i1	m1
p1	i1	m2

- Connaissant un projet et un ingénieur, il n'existe qu'un manager attribué

Associations ternaires : Many-Many-One

Modèle EA



Dépendance fonctionnelle

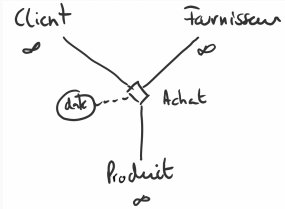
$\{id_projet, id_ingenieur\} \rightarrow \{id_manager\}$

Solution

Attribution(id_projet, id_ingenieur, id_manager)
 $id_projet \subseteq \text{Projet.id},$
 $id_ingenieur \subseteq \text{Ingénieur.id},$
 $id_manager \subseteq \text{Manager.id}$

Associations ternaires : Many-Many-Many

Modèle EA



Dépendance fonctionnelle (v1)

$\{\text{id_client, id_fournisseur, id_produit}\} \rightarrow \{\text{date}\}$

Interprétation

- Un client achète un produit qu'une fois auprès d'un même fournisseur à une date donnée

Solution

Attribution(id_client, id_fournisseur, id_produit, date)

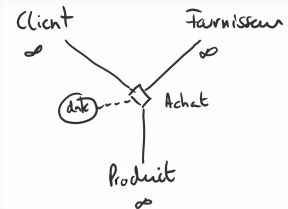
id_client \subseteq Client.id,

id_fournisseur \subseteq Fournisseur.id,

id_produit \subseteq Produit.id

Associations ternaires : Many-Many-Many

Modèle EA



Dépendance fonctionnelle (v2)

$\{id_achat\} \rightarrow \{id_cli, id_fourn, id_prod, date\}$

Solution

Attribution(id_achat, id_cli, id_fourn, id_prod, date)

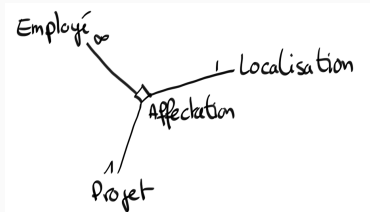
id_cli \subseteq Client.id,

id_fourn \subseteq Fournisseur.id,

id_prod \subseteq Produit.id

Associations ternaires : Many-One-One

Modèle EA



Dépendances fonctionnelles

$\{id_emp, id_loc\} \rightarrow \{id_proj\}$

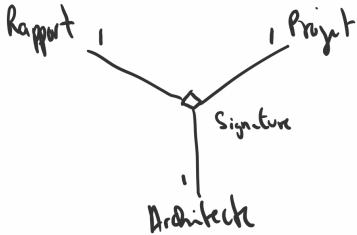
$\{id_emp, id_proj\} \rightarrow \{id_loc\}$

Solution

Affectation(id_emp, id_loc, id_proj)
(id_emp, id_proj) **UNIQUE**,
 $id_emp \subseteq Employe.id$,
 $id_loc \subseteq Localisation.id$,
 $id_proj \subseteq Projet.id$

Associations ternaires : One-One-One

Modèle EA



Dépendances fonctionnelles

$\{id_rapp, id_proj\} \rightarrow \{id_arch\}$

$\{id_rapp, id_arch\} \rightarrow \{id_proj\}$

$\{id_arch, id_proj\} \rightarrow \{id_rapp\}$

Solution

Signature(id_rapp, id_proj, id_arch)
(id_rapp, id_arch) **UNIQUE**,
(id_arch, id_proj) **UNIQUE**,
 $id_rapp \subseteq Rapport.id$,
 $id_proj \subseteq Projet.id$,
 $id_arch \subseteq Architecte.id$

- Les cardinalités sur les associations ternaires se décrivent facilement
- Sur des cas particuliers ou les associations de plus de trois entités, cela devient difficile
- Solution: décrire les **DF** en annotation du modèle EA
- Sur un modèle relationnelle, s'exprime facilement à l'aide de clés composées ou des contraintes d'unicité

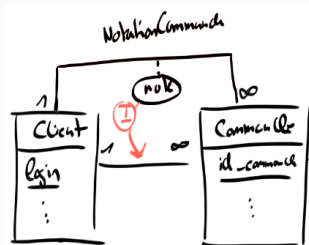
Dépendance d'inclusion et dépendance de jointure

Dépendance d'inclusion (DI)

- Une **DI** indique que les valeurs d'un ou de plusieurs attributs d'un enregistrement doivent exister dans une autre table
 - La **clé étrangère** est une **DI**
 - Une DI n'est pas nécessairement une clé étrangère
- Permet de renforcer (tout comme les DF) les contraintes d'intégrité
- Se modélise facilement

Dépendance d'inclusion (DI)

- Exemple: un client note une commande uniquement si la commande appartient au client



Client(login, prenom, ...)

Commande(id_commande, login, prix_unitaire, ...)

$\text{login} \subseteq \text{Client.login}$

NotationCommande(id_commande, login, note)

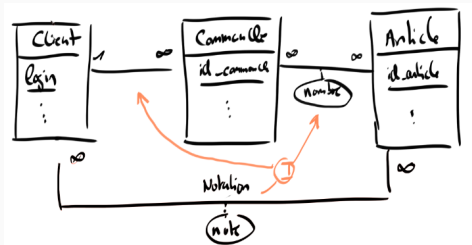
$(\text{login}, \text{id_commande}) \subseteq \text{Commande}(\text{login}, \text{id_commande})$

Dépendance de jointure (DJ)

- Une **DJ** est une **DI** qui s'opère sur des attributs de TE différentes
- En relationnel, difficilement modélisable:
 - se modélise via un trigger
 - ou par redondance d'une DF ou d'association n-aires (cf. exercice)

Dépendance de jointure (DJ)

- Exemple



Client(login, prenom, ...)

Commande(id_commande, login, prix_unitaire, ...)

$\text{login} \subseteq \text{Client.login}$

Article(id_article, prix_unitaire, ...)

CommandeArticle(id_article, id_commande, quantite, ...)

Notation(login, id_article, note)

$(\text{login}, \text{id_article}) \subseteq (\text{CommandeArticle} \times \text{Commande}).(\text{login}, \text{id_article})$

Cas d'utilisation 1

Cas d'utilisation 1

Dans cet exemple, rien ne garanti que le fournisseur choisi par le client pour son article fournit l'article en question.

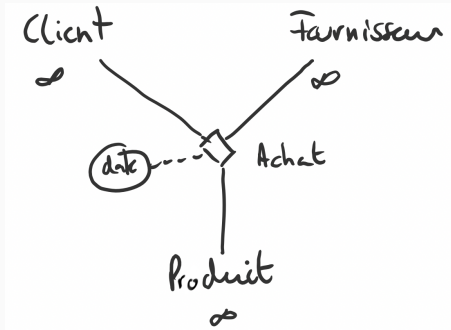


Figure 1: EA Achat

Cas d'utilisation 1

Ce modèle permet d'associer un produit à un fournisseur.

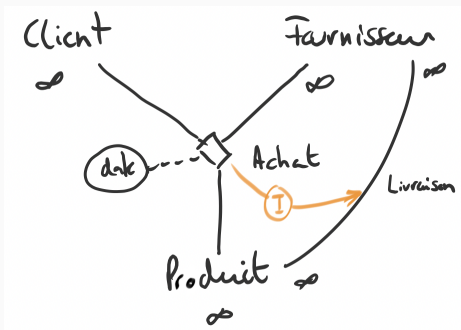


Figure 2: EA Achat avec livraison

Mais comment s'assurer que le produit qui se trouve sur un achat est bien associé à un fournisseur qui livre ce produit ?

Cas d'utilisation 1

Relations

Achat(id_achat, id_fourn, id_client, id_produit, date), ref: ...

Livraison(id_fourn, id_produit), ref: ...

Livraison

id_fourn	id_produit
F1	P1
F1	P2
F2	P1
F3	P4

Achat

id_achat	id_fourn	id_produit	id_client	date
A1	F2	P1	C2	d1
A2	F1	P4	C1	d2

Remarques

- F1 ne livre pourtant pas le produit P4

Cas d'utilisation 1

Relations

Achat(id_achat, id_fourn, id_client, id_produit, date), ref: ...
(id_fourn, id_produit) \subseteq Livraison.(id_fournisseur, id_produit)
Livraison(id_fournisseur, id_produit)

Livraison

id_fourn	id_produit
F1	P1
F1	P2
F2	P1
F3	P4

Achat

id_achat	id_fourn	id_produit	id_client	date
A1	F2	P1	C2	d1
A2	F1	P4	C1	d2

Cas d'utilisation 2

Cas d'utilisation 2

Raisonnons sur ce nouvel exemple

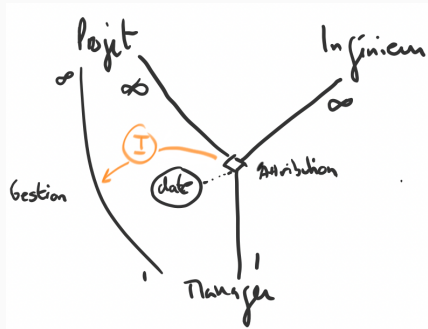


Figure 3: EA Attribution et Gestion

Quel est le modèle relationnel associé ?

Cas d'utilisation 2

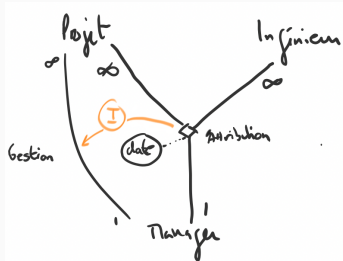


Figure 4: EA Attribution et Gestion

Proposition

Attribution(id_projet, id_ing, id_manager, date),
(id_projet, id_manager) \subseteq Projet.(id_projet, id_manager)
Projet(id_projet, ..., id_manager),
id_manager \subseteq Manager.id_manager
Manager(id_manager, ...)

Cas d'utilisation 2



Figure 5: EA Attribution et Gestion

Peut-on simplifier ?

Cas d'utilisation 2

Relation Attribution (sans clé)

Attribution(id_projet, id_ingenieur, id_manager, date)

Dépendances fonctionnelles associées

$\{id_projet, id_ingenieur\} \rightarrow \{id_manager, date\}$

$\{id_projet\} \rightarrow \{id_manager\}$

Etant donné que l'id du manager peut être déduit à partir de l'id du projet uniquement, on peut simplifier les DFs:

$\{id_projet, id_ingenieur\} \rightarrow \{date\}$

$\{id_projet\} \rightarrow \{id_manager\}$

Ce qui nous a permis de déduire les clés pour nos relations

Attribution(id_projet, id_ingenieur, date)

Projet(id_projet, id_manager)

Cas d'utilisation 2

Ce raisonnement nous aurait aidé à simplifier le modèle EA

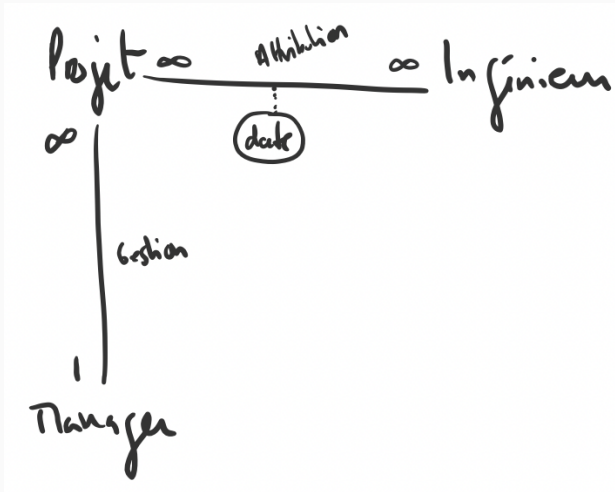


Figure 6: EA Attribution et Gestion

Conclusion sur les associations n-aires

- En connaissant les règles de DF il est possible de simplifier les modèles (cours d'alg. rel. chapitre 3)
- Il peut être utile de simplifier des associations n-aires en plusieurs associations binaires au risque de perdre des DF