D'un schéma conceptuel vers un schéma relationnel

Olivier Caron

Polytech Lille
Avenue Paul Langevin Cité Scientifique
Université de Lille
59655 Villeneuve d'Ascq cedex

http://ocaron.polytech-lille.net
Olivier.Caron@polytech-lille.fr

Problématique

- En entrée : on dispose d'un schéma conceptuel de données formalisé dans le langage UML.

Problématique

- En entrée : on dispose d'un schéma conceptuel de données formalisé dans le langage UML.
- En sortie : on veut obtenir un schéma relationnel le plus conforme possible au schéma conceptuel.

Problématique

- En entrée : on dispose d'un schéma conceptuel de données formalisé dans le langage UML.
- En sortie : on veut obtenir un schéma relationnel le plus conforme possible au schéma conceptuel.
- On parle de traduction ou bien de translation pour spécifier cette phase.

Les difficultés

- Des **concepts** distincts :

Les difficultés

- Des **concepts** distincts :
  - Dans la notation UML, on parle de classes, attributs d'une classe, associations entre classes (rôles, cardinalités min et max, navigation).

Les difficultés

- Des **concepts** distincts :
  - Dans la notation UML, on parle de classes, attributs d'une classe, associations entre classes (rôles, cardinalités min et max, navigation).
  - Dans le modèle relationnel, on parle de tables, colonnes, clés primaires et étrangères.

Les difficultés

- Des **concepts** distincts :
  - Dans la notation UML, on parle de classes, attributs d'une classe, associations entre classes (rôles, cardinalités min et max, navigation).
  - Dans le modèle relationnel, on parle de tables, colonnes, clés primaires et étrangères.
- Des restrictions différentes entre concepts équivalents.
  Exemple : le nom d'une **table** ne peut contenir que des caractères alphabétiques non accentués et chiffres. Le nom d'une **classe** UML est libre (caractères accentués, espaces,...)

Les difficultés

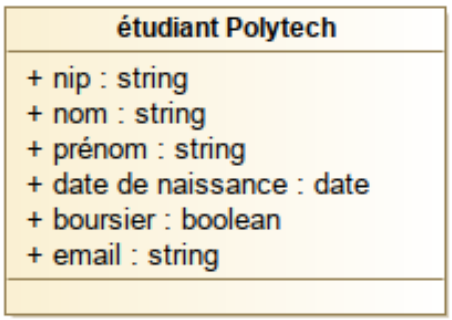
- Des **concepts** distincts :
  - Dans la notation UML, on parle de classes, attributs d'une classe, associations entre classes (rôles, cardinalités min et max, navigation).
  - Dans le modèle relationnel, on parle de tables, colonnes, clés primaires et étrangères.
- Des restrictions différentes entre concepts équivalents.
  Exemple : le nom d'une **table** ne peut contenir que des caractères alphabétiques non accentués et chiffres. Le nom d'une **classe** UML est libre (caractères accentués, espaces,. . .)
- On trouve des concepts similaires (ex : **table** et **classe**, **colonne** et **attribut**) mais ce n'est pas toujours possible, quelles sont les solutions possibles ?

Les difficultés

- Des **concepts** distincts :
  - Dans la notation UML, on parle de classes, attributs d'une classe, associations entre classes (rôles, cardinalités min et max, navigation).
  - Dans le modèle relationnel, on parle de tables, colonnes, clés primaires et étrangères.
- Des restrictions différentes entre concepts équivalents.
  Exemple : le nom d'une **table** ne peut contenir que des caractères alphabétiques non accentués et chiffres. Le nom d'une **classe** UML est libre (caractères accentués, espaces,. . .)
- On trouve des concepts similaires (ex : **table** et **classe**, **colonne** et **attribut**) mais ce n'est pas toujours possible, quelles sont les solutions possibles ?
- Parfois, plusieurs traductions sont possibles, laquelle choisir ?

Commençons par un exemple simple

- Traduire ce schéma UML en schéma relationnel



L'exemple simple : un bilan (1/4)

- Règle de traduction 1 : chaque classe de nom *X* devient une table de nom *X*
  Difficulté(s) :

L'exemple simple : un bilan (1/4)

- Règle de traduction 1 : chaque classe de nom *X* devient une table de nom *X*
  Difficulté(s) :
  - le nom de la table doit être éventuellement traduit pour être conforme à la norme SQL
    ex : "étudiant Polytech" $\Rightarrow$ "etudiantPolytech"

L'exemple simple : un bilan (1/4)

- Règle de traduction 1 : chaque classe de nom *X* devient une table de nom *X*
  Difficulté(s) :
  - le nom de la table doit être éventuellement traduit pour être conforme à la norme SQL
    ex : "étudiant Polytech" $\Rightarrow$ "etudiantPolytech"
- Règle de traduction 2 : un attribut *X* d'une classe *Y* devient une colonne *X* de la table *Y*
  Difficulté(s)

L'exemple simple : un bilan (1/4)

- Règle de traduction 1 : chaque classe de nom *X* devient une table de nom *X*
  Difficulté(s) :
  - le nom de la table doit être éventuellement traduit pour être conforme à la norme SQL
    ex : "étudiant Polytech" $\Rightarrow$ "etudiantPolytech"
- Règle de traduction 2 : un attribut *X* d'une classe *Y* devient une colonne *X* de la table *Y*
  Difficulté(s)
  - le nom de la colonne doit être éventuellement traduit pour être conforme à la norme SQL
    Exemples :"prénom" $\Rightarrow$ "prenom" et
    "date de naissance" $\Rightarrow$ "dateNaissance"

L'exemple simple : un bilan (2/4)

- Règle de traduction 3 : Fixer des traductions des types des attributs UML en types SQL
  ”integer” $\Rightarrow$ ”int”, ”double” $\Rightarrow$ ”float”, ”char” $\Rightarrow$ ”char”, ”boolean” $\Rightarrow$ ”boolean”, ”string” $\Rightarrow$ ”varchar(256)”,. . .
  Difficulté(s)

L'exemple simple : un bilan (2/4)

- Règle de traduction 3 : Fixer des traductions des types des attributs UML en types SQL
  ”integer” $\Rightarrow$ ”int”, ”double” $\Rightarrow$ ”float”, ”char” $\Rightarrow$ ”char”, ”boolean” $\Rightarrow$ ”boolean”, ”string” $\Rightarrow$ ”varchar(256)”,. . .
  Difficulté(s)
  - Plusieurs solutions : ”string” $\Rightarrow$ ”varchar(256)” ou ”text” ou . . .

L'exemple simple : un bilan (2/4)

- Règle de traduction 3 : Fixer des traductions des types des attributs UML en types SQL
  "integer" $\Rightarrow$ "int", "double" $\Rightarrow$ "float", "char" $\Rightarrow$ "char", "boolean" $\Rightarrow$ "boolean", "string" $\Rightarrow$ "varchar(256)",...
  Difficulté(s)
  - Plusieurs solutions : "string" $\Rightarrow$ "varchar(256)" ou "text" ou ...
  - Que faire des types SQL non connus en UML ? (ex : `timestamp`)

L'exemple simple : un bilan (2/4)

- Règle de traduction 3 : Fixer des traductions des types des attributs UML en types SQL
  "integer" $\Rightarrow$ "int", "double" $\Rightarrow$ "float", "char" $\Rightarrow$ "char", "boolean" $\Rightarrow$ "boolean", "string" $\Rightarrow$ "varchar(256)",...
  Difficulté(s)
  - Plusieurs solutions : "string" $\Rightarrow$ "varchar(256)" ou "text" ou ...
  - Que faire des types SQL non connus en UML ? (ex : `timestamp`)

L'exemple simple : un bilan (2/4)

- Règle de traduction 3 : Fixer des traductions des types des attributs UML en types SQL
  "integer" $\Rightarrow$ "int", "double" $\Rightarrow$ "float", "char" $\Rightarrow$ "char", "boolean" $\Rightarrow$ "boolean", "string" $\Rightarrow$ "varchar(256)",...
  Difficulté(s)
  - Plusieurs solutions : "string" $\Rightarrow$ "varchar(256)" ou "text" ou ...
  - Que faire des types SQL non connus en UML ? (ex : `timestamp`)
    Solution : définir des `DataType` au niveau UML pour intégrer les types SQL.

L'exemple simple : un bilan (3/4)

- Trouver une règle de traduction no 4 pour les clés primaires ”Toute table **doit** comporter une clé primaire”.

L'exemple simple : un bilan (3/4)

- Trouver une règle de traduction no 4 pour les clés primaires ”Toute table **doit** comporter une clé primaire”.
- Solution 1 : ajouter une colonne `id` de type `serial` pour chaque table.

L'exemple simple : un bilan (3/4)

- Trouver une règle de traduction no 4 pour les clés primaires
  "Toute table **doit** comporter une clé primaire".
- Solution 1 : ajouter une colonne `id` de type `serial` pour chaque table.
- Solution 2 : identifier une colonne qui peut faire office de clé primaire
  Pour cet exemple, la colonne "nip" peut jouer ce rôle.

L'exemple simple : un bilan (3/4)

- Trouver une règle de traduction no 4 pour les clés primaires
  "Toute table **doit** comporter une clé primaire".
- Solution 1 : ajouter une colonne `id` de type `serial` pour chaque table.
- Solution 2 : identifier une colonne qui peut faire office de clé primaire
  Pour cet exemple, la colonne "nip" peut jouer ce rôle.
- Difficultés :

L'exemple simple : un bilan (3/4)

- Trouver une règle de traduction no 4 pour les clés primaires
  "Toute table **doit** comporter une clé primaire".
- Solution 1 : ajouter une colonne `id` de type `serial` pour chaque table.
- Solution 2 : identifier une colonne qui peut faire office de clé primaire
  Pour cet exemple, la colonne "nip" peut jouer ce rôle.
- Difficultés :
  - Est-on certain de respecter l'analyse du concepteur UML ?

L'exemple simple : un bilan (3/4)

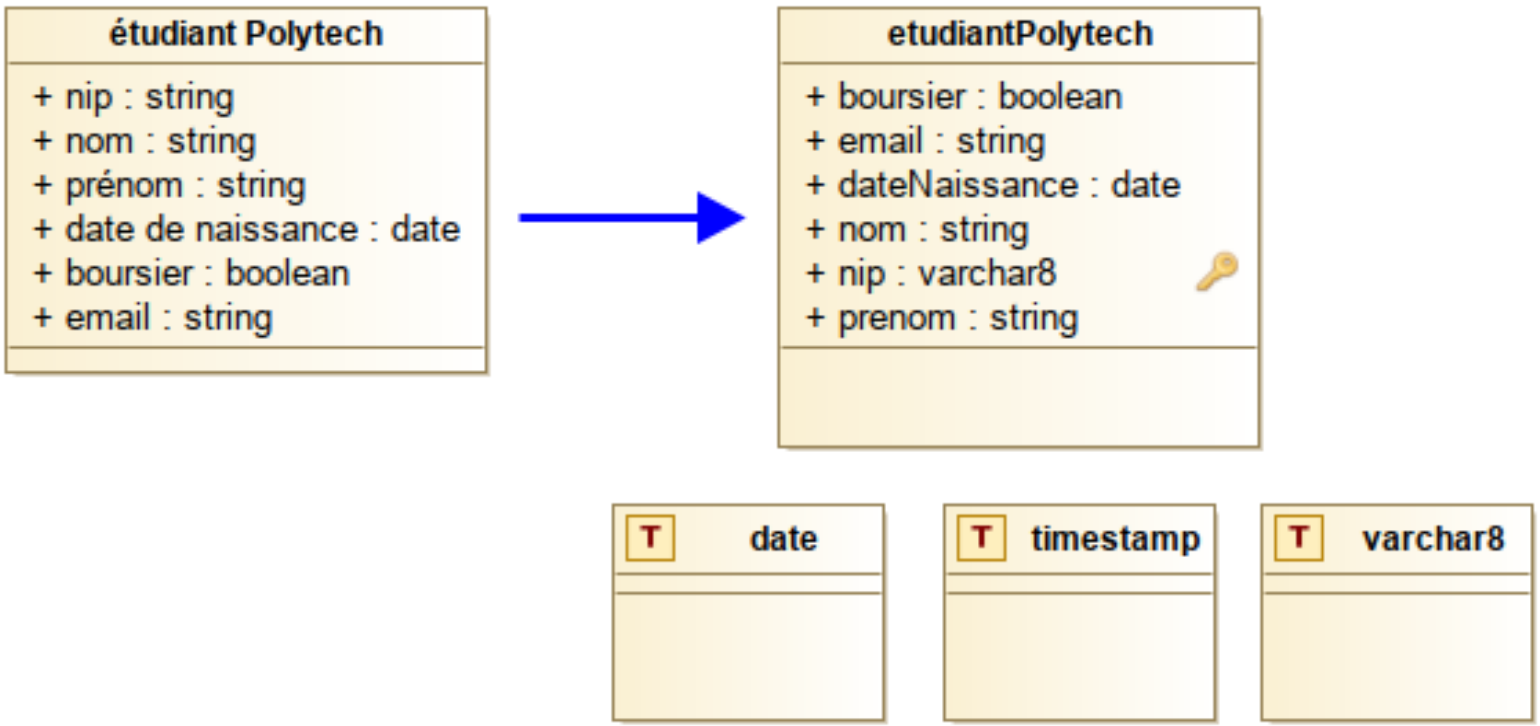
- Trouver une règle de traduction no 4 pour les clés primaires
  "Toute table **doit** comporter une clé primaire".
- Solution 1 : ajouter une colonne `id` de type `serial` pour chaque table.
- Solution 2 : identifier une colonne qui peut faire office de clé primaire
  Pour cet exemple, la colonne "nip" peut jouer ce rôle.
- Difficultés :
  - Est-on certain de respecter l'analyse du concepteur UML ?
  - Que faire quand aucune colonne ne peut jouer le rôle de clé primaire ?

L'exemple simple : un bilan (3/4)

- Trouver une règle de traduction no 4 pour les clés primaires
  "Toute table **doit** comporter une clé primaire".
- Solution 1 : ajouter une colonne `id` de type `serial` pour chaque table.
- Solution 2 : identifier une colonne qui peut faire office de clé primaire
  Pour cet exemple, la colonne "nip" peut jouer ce rôle.
- Difficultés :
  - Est-on certain de respecter l'analyse du concepteur UML ?
  - Que faire quand aucune colonne ne peut jouer le rôle de clé primaire ?
- Solution retenue : c'est le concepteur UML qui spécifie l'identifiant.
  Utilisation du mécanisme extensible d'UML, notion de stéréotype appliqué aux attributs UML.

L'exemple simple : un bilan (4/4)

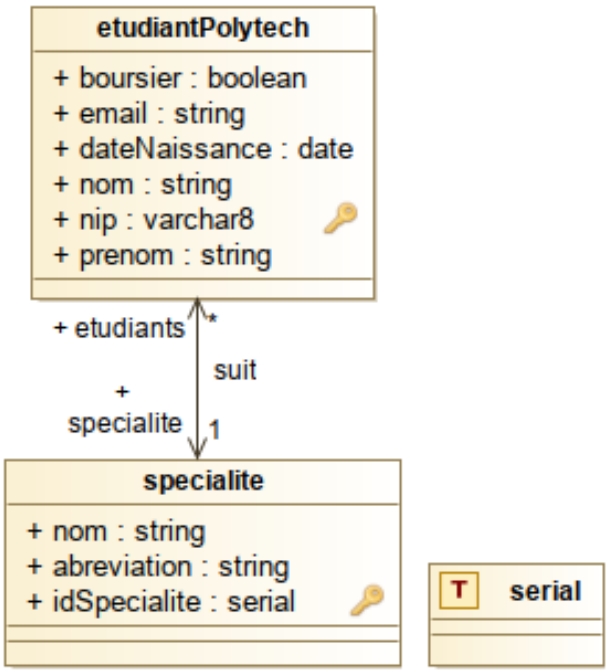
- Démarche pragmatique : le concepteur UML spécifie son schéma UML de manière à diminuer les alternatives de traduction :



...

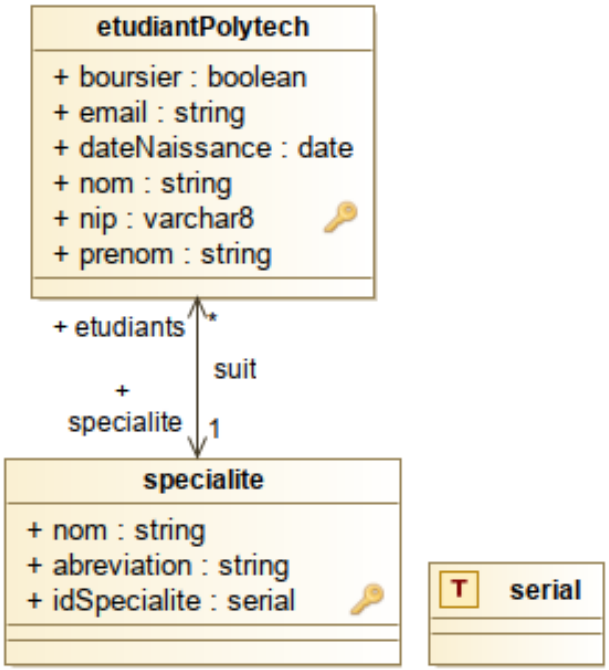
Poursuivons par un second exemple

- Traduire ce schéma UML en schéma relationnel



Poursuivons par un second exemple

- Traduire ce schéma UML
  en schéma relationnel



- Au fait, pourquoi ne pas avoir utilisé `nom` comme identifiant de la spécialité ?

Première solution : ”Une association devient une table”

- C’est la solution qui marche toujours (mais pas forcément la plus efficace)

Première solution : ”Une association devient une table”

- C’est la solution qui marche toujours (mais pas forcément la plus efficace)
  - L’association `a` devient la table `a` (moyennant caractères autorisés)

Première solution : ”Une association devient une table”

- C’est la solution qui marche toujours (mais pas forcément la plus efficace)
  - L’association `a` devient la table `a` (moyennant caractères autorisés)
  - Création de colonnes qui sont des clés étrangères pour désigner les tables issues des classes qui participent à l’association. Le nom de la colonne peut (devrait) contenir le nom du rôle de l’association.

Première solution : ”Une association devient une table”

- C’est la solution qui marche toujours (mais pas forcément la plus efficace)
  - L’association `a` devient la table `a` (moyennant caractères autorisés)
  - Création de colonnes qui sont des clés étrangères pour désigner les tables issues des classes qui participent à l’association. Le nom de la colonne peut (devrait) contenir le nom du rôle de l’association.
  - S’il existe des propriétés d’association alors création des colonnes correspondantes dans la table créée

Première solution : ”Une association devient une table”

- C’est la solution qui marche toujours (mais pas forcément la plus efficace)
  - L’association `a` devient la table `a` (moyennant caractères autorisés)
  - Création de colonnes qui sont des clés étrangères pour désigner les tables issues des classes qui participent à l’association. Le nom de la colonne peut (devrait) contenir le nom du rôle de l’association.
  - S’il existe des propriétés d’association alors création des colonnes correspondantes dans la table créée
  - Ce sont les cardinalités max qui vont définir la constitution de la clé primaire. Ici, un étudiant a au plus une spécialité, donc la clé est uniquement constitué de `refEtudiant`

Première solution : ”Une association devient une table”

- C’est la solution qui marche toujours (mais pas forcément la plus efficace)
  - L’association `a` devient la table `a` (moyennant caractères autorisés)
  - Création de colonnes qui sont des clés étrangères pour désigner les tables issues des classes qui participent à l’association. Le nom de la colonne peut (devrait) contenir le nom du rôle de l’association.
  - S’il existe des propriétés d’association alors création des colonnes correspondantes dans la table créée
  - Ce sont les cardinalités max qui vont définir la constitution de la clé primaire. Ici, un étudiant a au plus une spécialité, donc la clé est uniquement constitué de `refEtudiant`

etudiantPolytech(nip : varchar(8), boursier : boolean, email : varchar(300), dateNaissance :date, nom : varchar(200), nip :varchar(8))
specialite(nom :varchar(300), abreviation : varchar(4), idSpecialite :serial)
suit(#refEtudiant(etudiantPolytech) :varchar(8),#refSpecialite(specialite) :int)

Seconde solution : ”Une association devient des colonnes”

- Ne fonctionne que s’il existe une cardinalité max de 1 lors d’une association **binaire**

Seconde solution : ”Une association devient des colonnes”

- Ne fonctionne que s’il existe une cardinalité max de 1 lors d’une association **binaire**
  - Ajout d’une ”colonne clé étrangère” pour désigner la table distante.

Seconde solution : ”Une association devient des colonnes”

- Ne fonctionne que s’il existe une cardinalité max de 1 lors d’une association **binaire**
  - Ajout d’une ”colonne clé étrangère” pour désigner la table distante.
  - Si c’est une association ”1-1”, alors choix à faire sur l’une des deux tables.

Seconde solution : ”Une association devient des colonnes”

- Ne fonctionne que s’il existe une cardinalité max de 1 lors d’une association **binaire**
  - Ajout d’une ”colonne clé étrangère” pour désigner la table distante.
  - Si c’est une association ”1-1”, alors choix à faire sur l’une des deux tables.
  - S’il existe des propriétés d’association alors ajout des colonnes correspondantes en plus de la ”colonne clé étrangère”

Seconde solution : ”Une association devient des colonnes”

- Ne fonctionne que s’il existe une cardinalité max de 1 lors d’une association **binaire**
  - Ajout d’une ”colonne clé étrangère” pour désigner la table distante.
  - Si c’est une association ”1-1”, alors choix à faire sur l’une des deux tables.
  - S’il existe des propriétés d’association alors ajout des colonnes correspondantes en plus de la ”colonne clé étrangère”

etudiantPolytech(nip : varchar(8), boursier : boolean, email : varchar(300), dateNaissance :date, nom : varchar(200), nip :varchar(8), #refSpecialite(specialite) : int)

specialite(nom :varchar(300), abreviation : varchar(4), idSpecialite :serial)

Les autres cas

- Associations n-aire ($n > 2$)

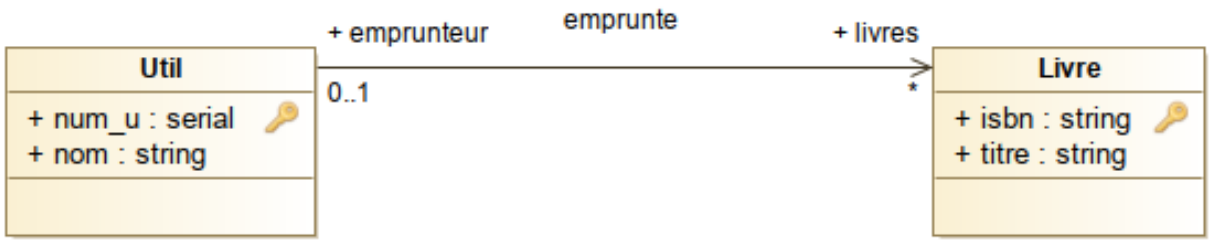
Les autres cas

- Associations n-aire ($n > 2$)
- Associations ”*-*”

Les autres cas

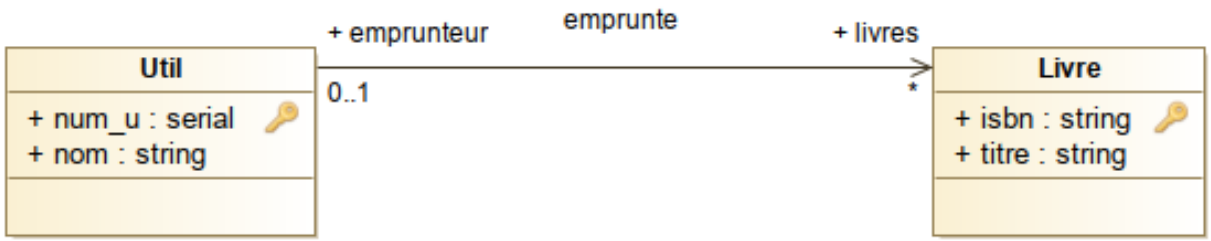
- Associations n-aire ($n > 2$)
- Associations ”*-*”
- Pour ces deux précédents cas, seule la solution ”Une association devient une table” est possible

Appliquons nos connaissances



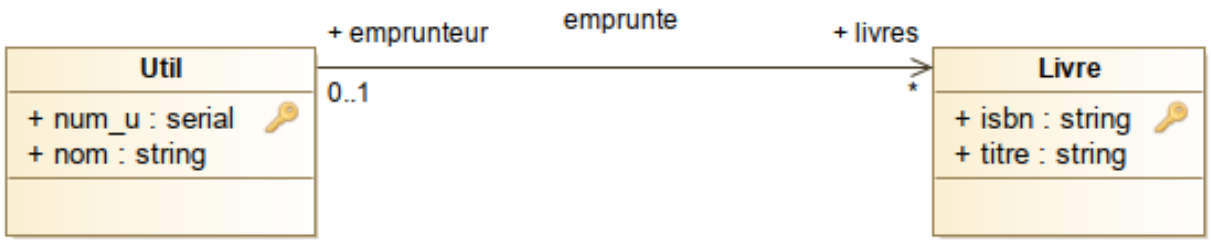
- Traduire ce schéma UML
  en schéma relationnel en utilisant la solution ”Une association devient une table”

Appliquons nos connaissances



- Traduire ce schéma UML
  en schéma relationnel en utilisant la solution ”Une association devient une table”
- Traduire ce schéma UML
  en schéma relationnel en utilisant la solution ”Une association devient des colonnes”

Appliquons nos connaissances



- Traduire ce schéma UML
  en schéma relationnel en utilisant la solution ”Une association devient une table”
- Traduire ce schéma UML
  en schéma relationnel en utilisant la solution ”Une association devient des colonnes”
- Laquelle a votre préférence ?

Résultat (1/2)

- la solution ”Une association devient une table” :

Résultat (1/2)

- la solution ”Une association devient une table” :

  Util(num_u : serial, nom : varchar(300))
  livre(isbn : varchar(13), titre : varchar(300))
  emprunte(#emprunteur(Util) : int, #refLivre(livre) : varchar(13))

Résultat (1/2)

- la solution ”Une association devient une table” :

  Util(num_u : serial, nom : varchar(300))
  livre(isbn : varchar(13), titre : varchar(300))

  emprunte(#emprunteur(Util) : int, #refLivre(livre) : varchar(13))

- nom colonne `emprunteur` plus explicite que `refUtil`

Résultat (1/2)

- la solution ”Une association devient une table” :

  Util(num_u : serial, nom : varchar(300))
  livre(isbn : varchar(13), titre : varchar(300))

  emprunte(#emprunteur(Util) : int, #refLivre(livre) : varchar(13))

- nom colonne `emprunteur` plus explicite que `refUtil`
- Attention, à la compatibilité des types clés primaires et clés étrangères

Résultat (1/2)

- la solution ”Une association devient une table” :

  Util(num_u : serial, nom : varchar(300))
  livre(isbn : varchar(13), titre : varchar(300))
  emprunte(#emprunteur(Util) : int, #refLivre(livre) : varchar(13))

- nom colonne `emprunteur` plus explicite que `refUtil`
- Attention, à la compatibilité des types clés primaires et clés étrangères
- La clé primaire de `emprunte` est constituée de `refLivre`.

Résultat (2/2)

- la solution ”Une association devient des colonnes” :

Résultat (2/2)

- la solution ”Une association devient des colonnes” :

  Util(num_u : serial, nom : varchar(300))
  livre(isbn : varchar(13), titre : varchar(300), #emprunteur(Util) : int)

Résultat (2/2)

- la solution ”Une association devient des colonnes” :

  Util(num_u : serial, nom : varchar(300))
  livre(isbn : varchar(13), titre : varchar(300), #emprunteur(Util) : int)

- Grâce à cette solution, on ”gagne une table”.

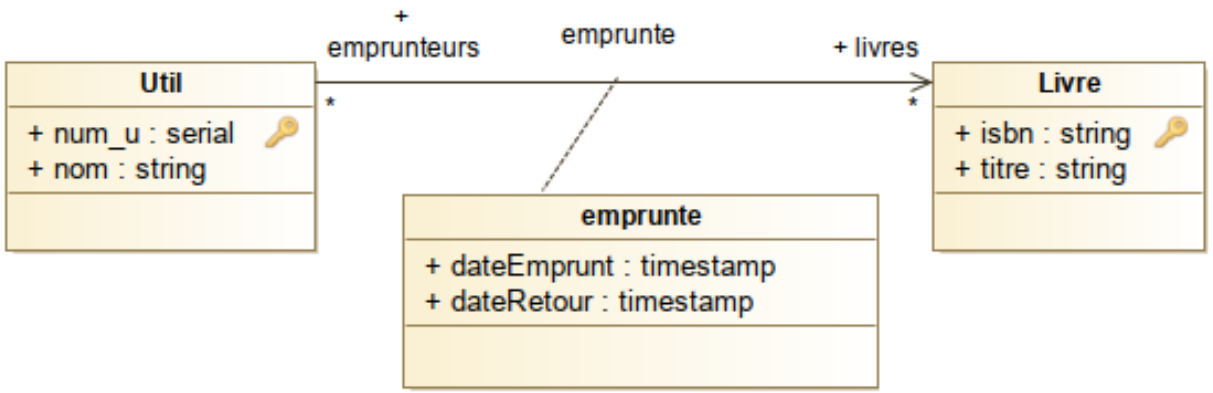
Résultat (2/2)

- la solution ”Une association devient des colonnes” :

  Util(num_u : serial, nom : varchar(300))
  livre(isbn : varchar(13), titre : varchar(300), #emprunteur(Util) : int)

- Grâce à cette solution, on ”gagne une table”.
- Mais dans une bibliothèque, une infirme partie des livres sont à un moment empruntés, la colonne `emprunteur` est majoritairement positionnée à `NULL`
  La précédente solution est plus efficace en terme d’espace disque.

Un dernier exercice



- Traduire ce schéma UML en schéma relationnel

Un dernier exercice - solution

- Ici, l'association emprunte est une association "*-*" $\Rightarrow$ seule la solution "Une association devient une table" est possible :

Un dernier exercice - solution

- Ici, l'association emprunte est une association "*-*" $\Rightarrow$ seule la solution "Une association devient une table" est possible :

  Util(num_u : serial, nom : varchar(300))
  livre(isbn : varchar(13), titre : varchar(300))
  emprunte(#emprunteur(Util) : int, #refLivre(livre) : varchar(13), dateEmprunt : timestamp, dateretour : timestamp)

Un dernier exercice - solution

- Ici, l'association emprunte est une association "*-*" $\Rightarrow$ seule la solution "Une association devient une table" est possible :

  Util(num_u : serial, nom : varchar(300))
  livre(isbn : varchar(13), titre : varchar(300))
  emprunte(#emprunteur(Util) : int, #refLivre(livre) : varchar(13), dateEmprunt : timestamp, dateretour : timestamp)

- Ici, la colonne `dateEmprunt` fait partie de la clé primaire car sinon un emprunteur ne peut plus ré-emprunter un même livre.

Un dernier exercice - solution

- Ici, l'association emprunte est une association "*-*" $\Rightarrow$ seule la solution "Une association devient une table" est possible :

  Util(num_u : serial, nom : varchar(300))
  livre(isbn : varchar(13), titre : varchar(300))
  emprunte(#emprunteur(Util) : int, #refLivre(livre) : varchar(13), dateEmprunt : timestamp, dateretour : timestamp)

- Ici, la colonne `dateEmprunt` fait partie de la clé primaire car sinon un emprunteur ne peut plus ré-emprunter un même livre.
- Le concepteur UML aurait pu annoter l'attribut `dateEmprunt` avec une clé.

Conclusion

- Préférable que le concepteur UML fournisse un maximum d'informations (noms des rôles, clés, règles de gestion, . . .)

Conclusion

- Préférable que le concepteur UML fournisse un maximum d'informations (noms des rôles, clés, règles de gestion, . . .)
- Dans un souci de traçabilité, ne pas changer les noms des éléments traduits.

Conclusion

- Préférable que le concepteur UML fournisse un maximum d'informations (noms des rôles, clés, règles de gestion, . . .)
- Dans un souci de traçabilité, ne pas changer les noms des éléments traduits.
- Il existe des outils automatiques de traduction UML vers Relationnel mais la solution obtenue n'est pas forcément la plus efficace.