# BDBIO - Modélisation relationnelle - niveau logique

Fabien Duchateau

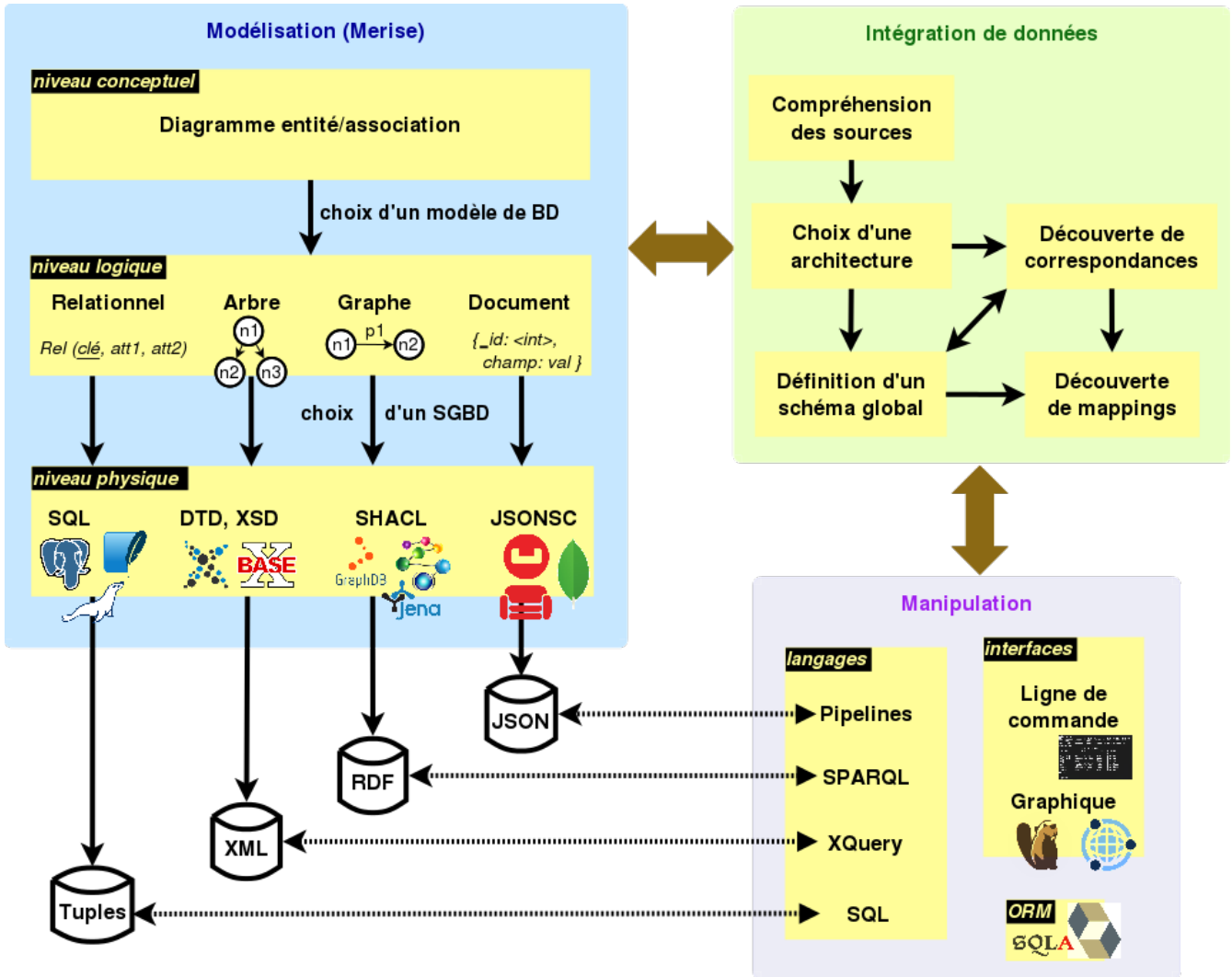
*fabien.duchateau [at] univ-lyon1.fr*

Université Claude Bernard Lyon 1

2024 - 2025


https://perso.liris.cnrs.fr/fabien.duchateau/BDBIO/

# Aperçu des notions de BDBIO




Ces diapositives utilisent le genre féminin (e.g., chercheuse, développeuses) plutôt que l'écriture inclusive (moins accessible, moins concise, et pas totalement inclusive)

# Rappels

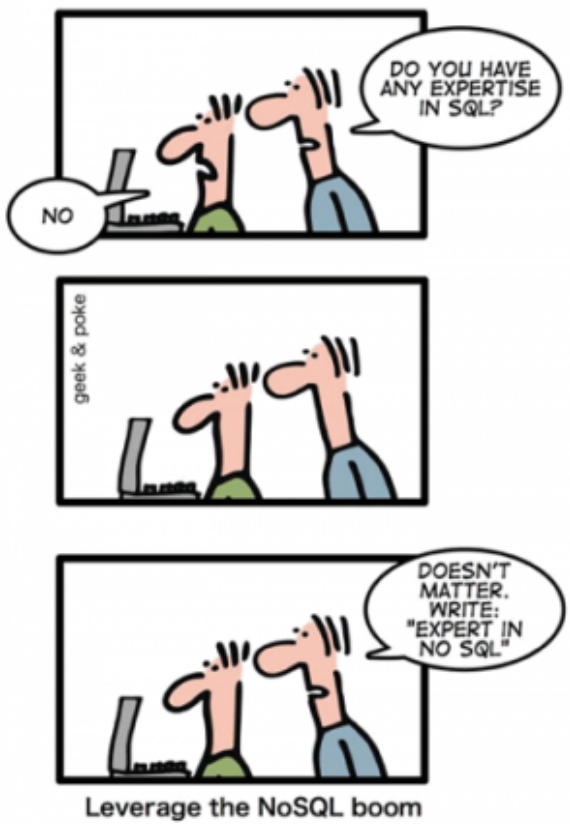
Un modèle conceptuel avec un niveau d'abstraction élevé :

- Communication et validation avec les clientes / utilisatrices
- Diagramme entité / association
  - entités (représentations des objets du domaine)
  - associations (liens entre entités)
  - cardinalité (pour une entité, son nombre de participations à une association)
  - des extensions (entité faible, spécialisation, etc.)

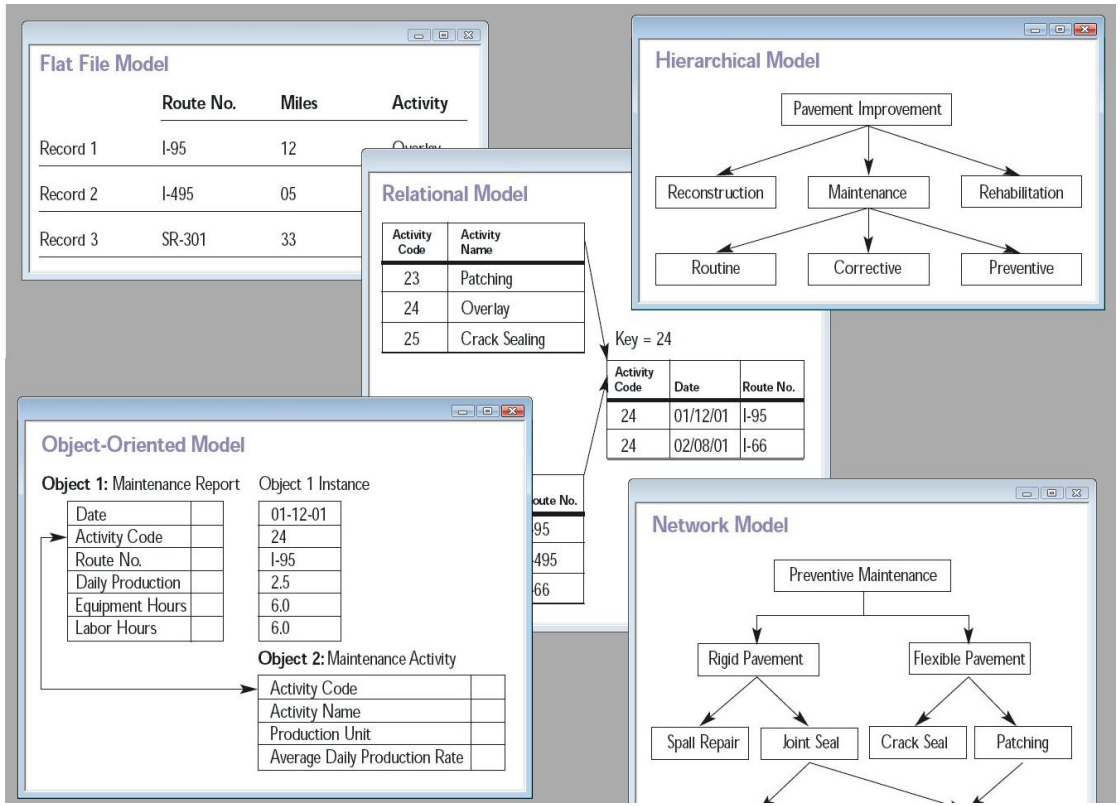
# Modèle logique

Le modèle logique va permettre de transformer les concepts du modèle conceptuel afin de se rapprocher d'un type de BD parmi :

- Hiérarchique : uniquement des liens 1-1 ou 1-n (arborescence depuis la racine)
- Réseau : ajout de liens n-m (représentation en graphe)
- Objet : collections d'objets (informations et traitements)
- Relationnel : collections de relations
- ...



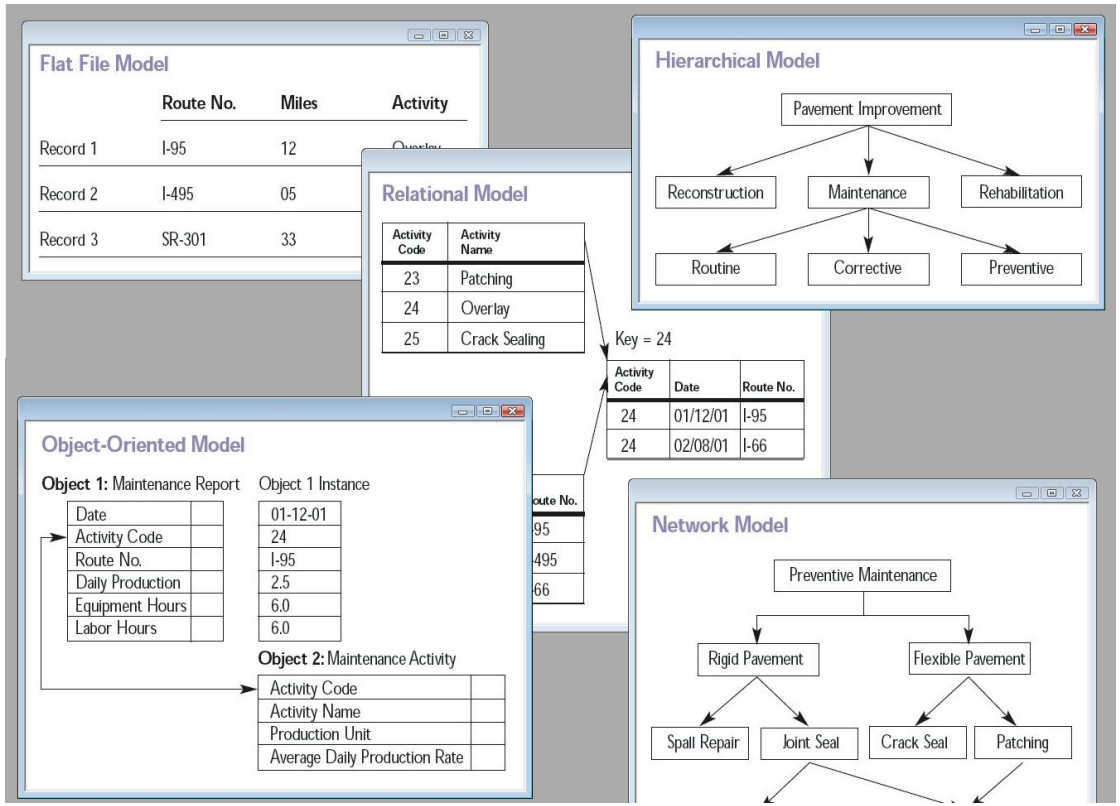
# Modèle logique - types de BD




http://en.wikipedia.org/wiki/Database_model

http://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es_relationnelle

# Modèle logique - types de BD




Le modèle logique des SGBD relationnels est le modèle relationnel

---

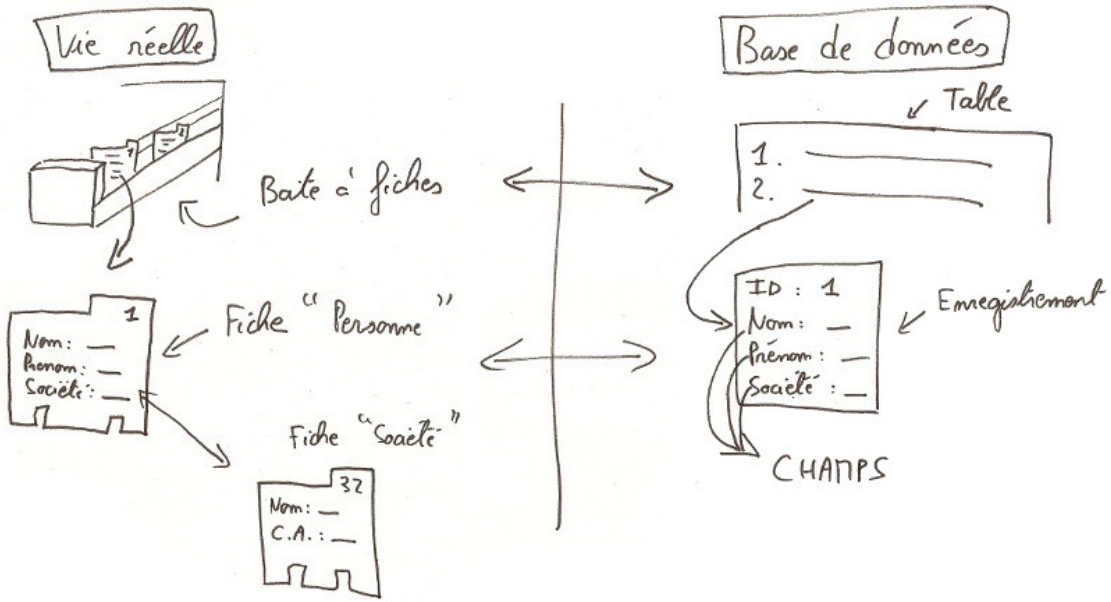
http://en.wikipedia.org/wiki/Database_model

http://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es_relationnelle

# Plan

# Généralités

- Défini en 1970 par Codd, modèle très populaire depuis
- Niveau d'abstraction plus élevé que les autres modèles logiques (hiérarchique et réseau)




Codd, Edgar. A relational model of data for large shared data banks. Communications of the ACM (1970)

# Avantages du modèle relationnel

Un modèle fondamentalement simple :

- Plus facilement compréhensible
- Plus facilement optimisable

Mais assez expressif :

- Possibilité de représenter des objets plus complexes

Utilisé en pratique depuis les années 80, implémenté dans de nombreux SGBD :

- Oracle, MySQL, MariaDB, PostgreSQL, DB2, SQL Server, ...

---

http://db-engines.com/en/ranking

# Concepts du modèle relationnel

Le modèle relationnel est un modèle ensembliste :

- Les objets sont **simples**, atomiques
  - entier, flottants, chaînes de caractères, dates, ...
  - pas de listes, pas de tableaux, pas de structures
- Opérations ensemblistes usuelles
  - union, intersection, différence, produit cartésien
- On utilise les **relations** pour représenter et manipuler les données selon la vision ensembliste
  - une relation portant sur $n$ ensembles $E_1,...,E_n$ est un sous-ensemble du produit cartésien $E_1 \times ... \times E_n$

Une instance de modèle relationnel = schéma relationnel + n-uplets / instances de relation

# Schéma relationnel

Un **schéma relationnel** est composé :

- D'un ensemble d'**attributs** :
  - décrit les données atomiques que l'on veut manipuler (e.g., un nom d'université, une ville)
- D'un ensemble de **relations** ou **tables** sur ces attributs :
  - représente les liens entre les données atomiques
  - permet de représenter des objets complexes
  - **arité** d'une relation : nombre d'attributs de cette relation

ÉLÈVE (idE, *nomE*, *moyenneLycée*, *effectifLycée*)
UNIVERSITÉ (nomU, *ville*, *effectif*)

# Schéma relationnel (2)

Un schéma relationnel inclut également des **contraintes** :

- Le type des attributs

nomU : `varchar`, ville : `varchar`, effectif : `integer`

- Domaine de valeurs = ensemble d'instances d'un type élémentaire (e.g., les entiers, les réels, chaîne de caractères)

- Des contraintes plus complexes

"L'effectif d'une université doit être supérieur à 0"

# Clés primaires et étrangères

- **Clé primaire** (Primary Key, PK) : le ou les attributs qui forment la clé primaire d'une relation permettent d'identifier sans ambiguité une instance de cette relation et sont soulignés dans le schéma relationnel

- **Clé étrangère** (Foreign Key, FK) : un attribut clé étrangère est une référence vers la clé primaire d'une autre table. Les valeurs de cet attribut doivent exister dans la table référencée. Dans le schéma relationnel, une clé étrangère est précédée d'un #dièse

# Un moment de réflexion

ÉLÈVE (idE, *nomE*, *moyenneLycée*, *effectifLycée*)
CANDIDATURE (#*idE*, #*nomU*, département, *décision*)
UNIVERSITÉ (nomU, *ville*, *effectif*)

- Combien de relations (tables) dans cette base de données ?

# Un moment de réflexion

| | |
|---|---|
| ÉLÈVE | (idE, *nomE*, *moyenneLycée*, *effectifLycée*) |
| CANDIDATURE | (#*idE*, #*nomU*, département, *décision*) |
| UNIVERSITÉ | (nomU, *ville*, *effectif*) |

- Combien de relations (tables) dans cette base de données ?
- Quelle est l'arité de la relation ÉLÈVE ?

# Un moment de réflexion

| | |
|---|---|
| ÉLÈVE | (idE, *nomE*, *moyenneLycée*, *effectifLycée*) |
| CANDIDATURE | (#*idE*, #*nomU*, département, *décision*) |
| UNIVERSITÉ | (nomU, *ville*, *effectif*) |

- Combien de relations (tables) dans cette base de données ?
- Quelle est l'arité de la relation ÉLÈVE ?
- Quelles sont les clés de ces relations ?

# Un moment de réflexion

ÉLÈVE (idE, *nomE*, *moyenneLycée*, *effectifLycée*)
CANDIDATURE (#*idE*, #*nomU*, département, *décision*)
UNIVERSITÉ (nomU, *ville*, *effectif*)

- Combien de relations (tables) dans cette base de données ?
- Quelle est l'arité de la relation ÉLÈVE ?
- Quelles sont les clés de ces relations ?
- Quel est (probablement) le domaine de *moyenneLycee* ?

# Un moment de réflexion

| | |
|---|---|
| ÉLÈVE | (idE, *nomE*, *moyenneLycée*, *effectifLycée*) |
| CANDIDATURE | (#*idE*, #*nomU*, département, *décision*) |
| UNIVERSITÉ | (nomU, *ville*, *effectif*) |

- Combien de relations (tables) dans cette base de données ?
- Quelle est l'arité de la relation ÉLÈVE ?
- Quelles sont les clés de ces relations ?
- Quel est (probablement) le domaine de *moyenneLycee* ?
- Quel type de contrainte peut-on imaginer sur l'attribut *décision* ?

# Instances

Une **instance d'une base de données** est un ensemble d'instances de relations (une instance par relation).

Une **instance de relation** $R(A_1,...,A_n)$ est un sous-ensemble fini du produit cartésien des domaines de ses attributs :

- Soit $D_1$ le domaine (du type) de $A_1$, ..., et $D_n$ le domaine (du type) de $A_n$, alors toute instance de R est incluse dans le produit cartésien $D_1 \times ... \times D_n$

| **nomU** | **ville** | **effectif** |
|---|---|---|
| INSA | Lyon | 36000 |
| UCB | Lyon | 15000 |
| UJF | Grenoble | 10000 |
| UJM | Saint-Étienne | 21000 |

Table UNIVERSITÉ

# Instances (2)

Les instances des relations servent à représenter les données. Ce sont ces instances de la relation qui sont stockées

L'instance est un ensemble de **n-uplets** (tuples en anglais)

{ (INSA, Lyon, 36000),
(UCB, Lyon, 15000),
(UJF, Grenoble, 10000),
(UJM, Saint-Étienne, 21000) }

# Instances (3)

Vision ensembliste des relations :

- L'ordre est éléments (tuples) n'est pas important
- Absence de doublons parmi les tuples
- Toutes les valeurs des attributs dans l'instance sont connues

# Instances (3)

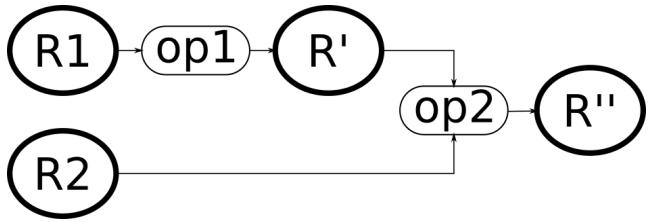
Vision ensembliste des relations :

- L'ordre est éléments (tuples) n'est pas important
- Absence de doublons parmi les tuples
- Toutes les valeurs des attributs dans l'instance sont connues

En pratique, les choses sont différentes !

# Manipulation des données

- Opérations pour interroger les (instances de) relations
  - Entrée : une ou plusieurs relations
  - Sortie : une relation




- Types d'opérations :
  - ensemblistes classiques = union, intersection, produit cartésien, etc.
  - spécifiques bases de données = sélection de n-uplets intéressants, projection d'attributs intéressants, etc.

# Exemple de BD relationnelle

ÉLÈVE (idE, *nomE*, *moyenneLycée*, *effectifLycée*)
CANDIDATURE (#*idE*, #*nomU*, département, *décision*)
UNIVERSITÉ (nomU, *ville*, *effectif*)

| idE | nomE | moyenneLycée | effectifLycée |
|---|---|---|---|
| 123 | Ana | 19.5 | 1000 |
| 234 | Bob | 18 | 1500 |
| 345 | Chloé | 17.5 | 500 |
| 456 | Damien | 19.5 | 1000 |
| 543 | Chloé | 17 | 2000 |
| 567 | Éléonore | 14.5 | 2000 |
| 654 | Ana | 19.5 | 1000 |
| 678 | Farid | 19 | 200 |
| 765 | Joana | 14.5 | 1500 |
| 789 | Gisèle | 17 | 800 |
| 876 | Irène | 19.5 | 400 |
| 898 | Hector | 18.5 | 800 |

Table ÉLÈVE

| nomU | ville | effectif |
|---|---|---|
| INSA | Lyon | 36000 |
| UCB | Lyon | 15000 |
| UJF | Grenoble | 10000 |
| UJM | Saint-Étienne | 21000 |

Table UNIVERSITÉ

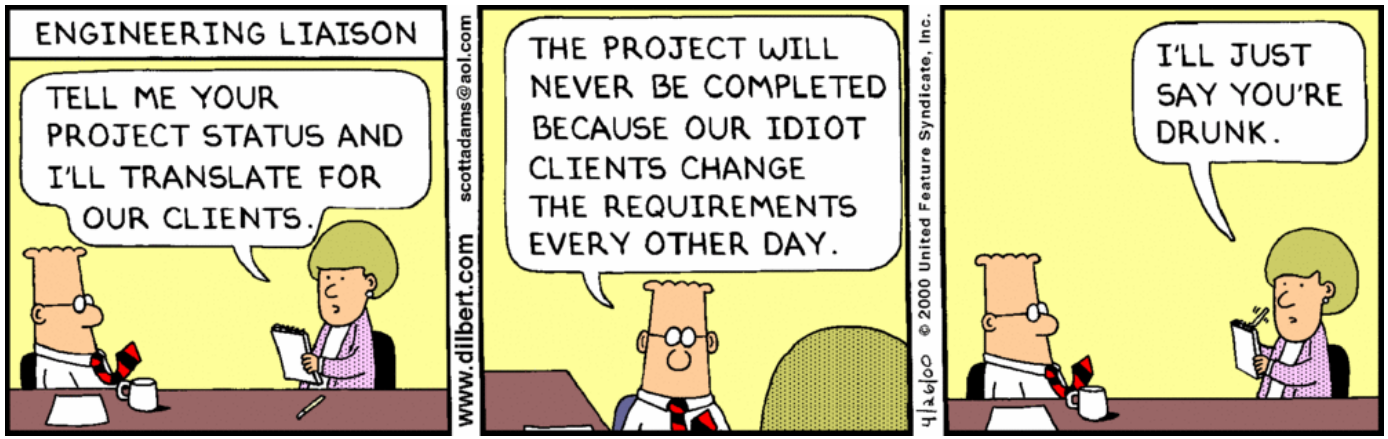
| idE | nomU | département | décision |
|---|---|---|---|
| 123 | INSA | informatique | O |
| 123 | UCB | électronique | N |
| 123 | UCB | informatique | O |
| 123 | UJM | électronique | O |
| 234 | INSA | biologie | N |
| 345 | UJF | bioinformatique | O |
| 345 | UJM | bioinformatique | N |
| 345 | UJM | électronique | N |
| 345 | UJM | informatique | O |
| 543 | UJF | informatique | N |
| 678 | UCB | histoire | O |
| 765 | UCB | histoire | O |
| 765 | UJM | histoire | N |
| 765 | UJM | psychologie | O |
| 876 | UCB | informatique | N |
| 876 | UJF | biologie | O |
| 876 | UJF | biologie marine | N |
| 898 | INSA | informatique | O |
| 898 | UCB | informatique | O |

Table CANDIDATURE

# En résumé

Dans un SGBD relationnel :

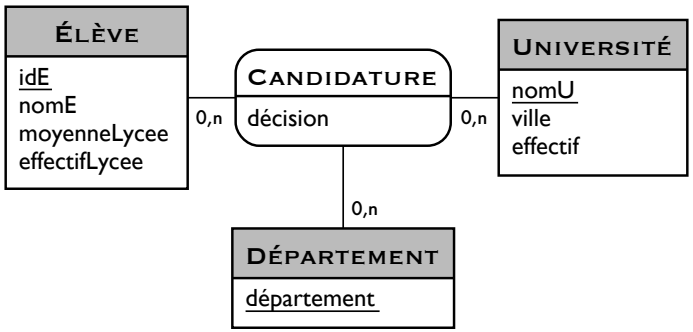
- Un ensemble de relations (ou tables)
- Schéma relationnel (relations, attributs typés, contraintes)
- Instance d'une relation (ensemble des n-uplets)



# Plan

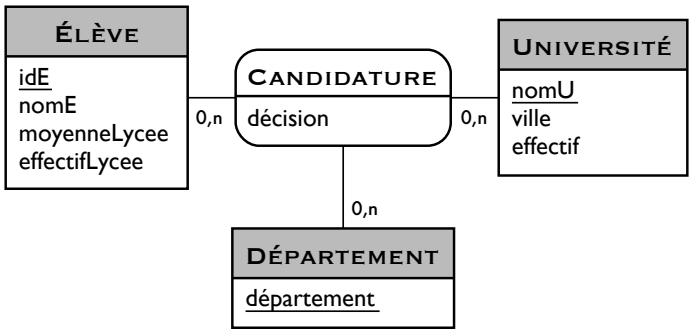
# Motivation

Comment transformer un diagramme E/A en schéma relationnel ?



# Motivation

Comment transformer un diagramme E/A en schéma relationnel ?




ÉLÈVE (idE, *nomE*, *moyenneLycée*, *effectifLycée*)
CANDIDATURE (#*idE*, #*nomU*, département, *décision*)
UNIVERSITÉ (nomU, *ville*, *effectif*)

# Règles de transformation E/A vers relationnel

Trois règles à appliquer pour le diagramme E/A de base :

1. Transformation des entités
2. Transformation des associations 0,1 ou 1,1
3. Transformation des autres associations

# Règle 1 - transformation des entités

- **Une entité devient une relation**
- Le ou les identifiants de l'entité deviennent clé primaire de la relation
- Les propriétés de l'entité deviennent attributs de la relation

# Règle 1 - transformation des entités

- **Une entité devient une relation**
- Le ou les identifiants de l'entité deviennent clé primaire de la relation
- Les propriétés de l'entité deviennent attributs de la relation

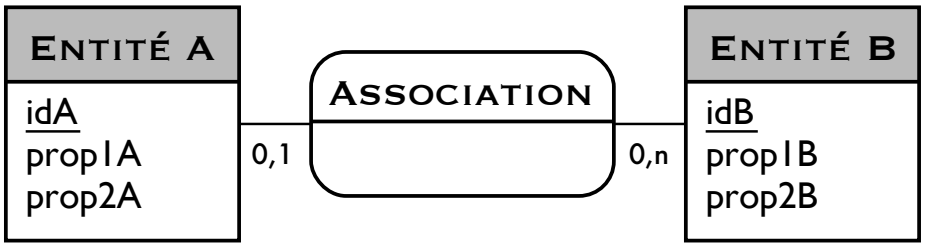

EntitéA(idA, prop1A, prop2A)

# Règle 2 - transformation des associations 0,1 ou 1,1

- **La relation correspondant à l'entité de cardinalité 0,1 ou 1,1 récupère comme attribut le ou les identifiants de l'autre entité participant à l'association**
- Ces attributs sont des clés étrangères (Foreign Key, FK), i.e., une référence vers l'identifiant d'une autre table. Les valeurs de cet attribut doivent exister dans la table référencée

# Règle 2 - transformation des associations 0,1 ou 1,1

- **La relation correspondant à l'entité de cardinalité 0,1 ou 1,1 récupère comme attribut le ou les identifiants de l'autre entité participant à l'association**
- Ces attributs sont des clés étrangères (Foreign Key, FK), i.e., une référence vers l'identifiant d'une autre table. Les valeurs de cet attribut doivent exister dans la table référencée




EntitéA(idA, prop1A, prop2A, **#idB**)
EntitéB(idB, prop1B, prop2B)

# Règle 3 - transformation des autres associations

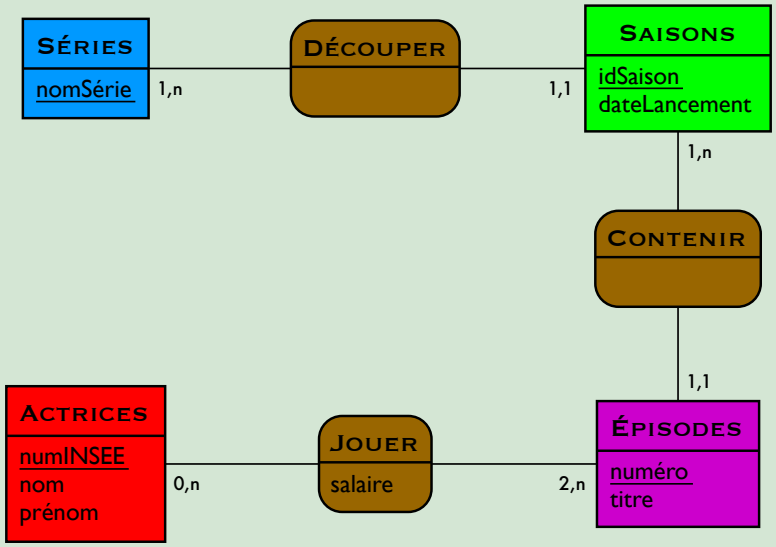
- **L'association devient une relation**
- Les identifiants des entités participant à l'association sont ajoutés comme clé primaire de cette relation (clé composée)
- Les propriétés de l'association deviennent des attributs de cette relation

# Règle 3 - transformation des autres associations

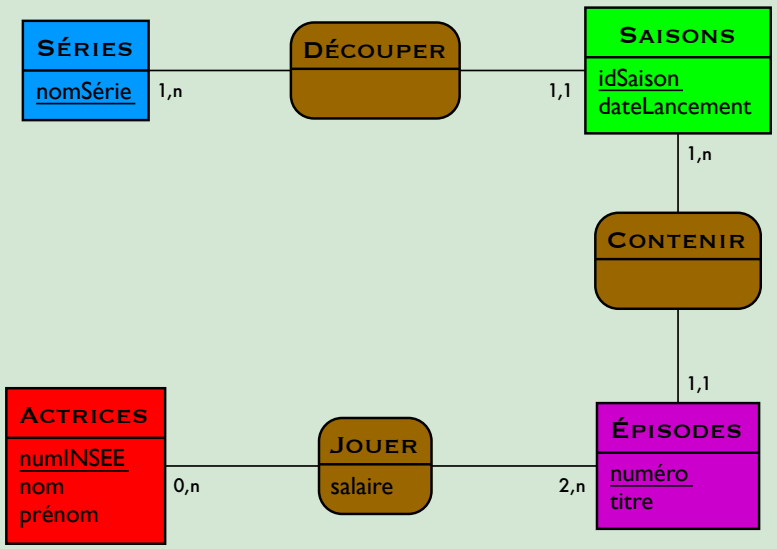
- **L'association devient une relation**
- Les identifiants des entités participant à l'association sont ajoutés comme clé primaire de cette relation (clé composée)
- Les propriétés de l'association deviennent des attributs de cette relation


EntitéA(idA, prop1A, prop2A)
EntitéB(idB, prop1B, prop2B)
Association(#idA, #idB, propAsso)

# Exercice - transformation E/A vers relationnel



# Exercice - transformation E/A vers relationnel




SÉRIES (nomSérie)
SAISONS (idSaison, dateLancement, #nomSérie)
ÉPISODES (numéro, titre, #idSaison)
ACTRICES (numINSEE, nom, prénom)
JOUER (#numéro, #numINSEE, salaire)

# En résumé

Trois règles simples à appliquer pour le modèle de base :

- Entité→ relation
- Association 0,1 ou 1,1→ clé étrangère
- Autres associations→ relation

# Plan

# Transformation du modèle étendu

Les règles de base permettent de transformer une grande partie d'un diagramme E/A

Mais il faut faire attention :

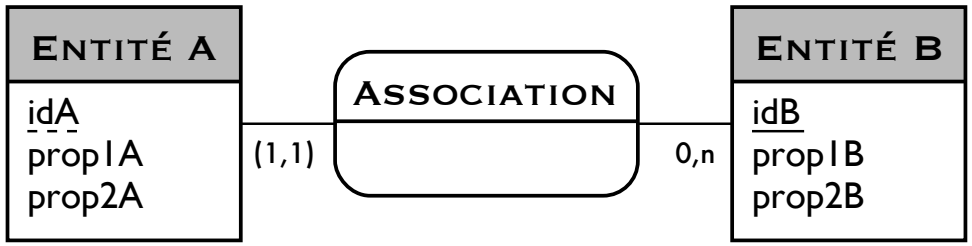
- Aux cas particuliers, notamment si l'on utilise le modèle E/A étendu (e.g., entité faible, agrégation)
- Aux exceptions aux trois règles de base (souvent dictées par le contexte)

# Transformation des entités faibles

- **L'entité faible devient une relation**
- Sa clé primaire se compose des identifiants de l'entité faible ainsi que des identifiants de l'entité forte (ces derniers étant aussi clés étrangères)
- Les propriétés de l'entité faible deviennent des attributs dans sa relation

# Transformation des entités faibles

- **L'entité faible devient une relation**
- Sa clé primaire se compose des identifiants de l'entité faible ainsi que des identifiants de l'entité forte (ces derniers étant aussi clés étrangères)
- Les propriétés de l'entité faible deviennent des attributs dans sa relation




EntitéA(idA, prop1A, prop2A, **#idB**)
EntitéB(idB, prop1B, prop2B)

# Transformation des spécialisations/généralisations

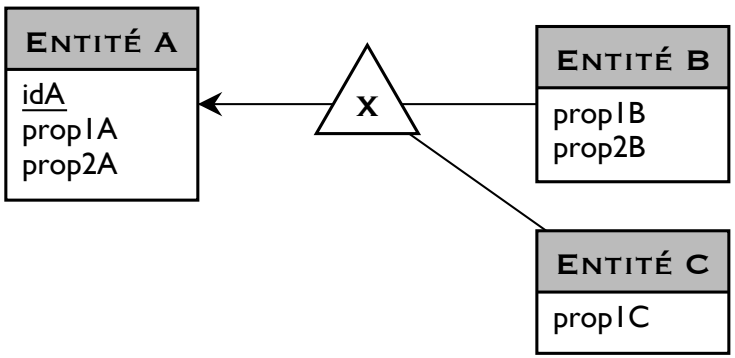
**Première solution : transformer le sur-type (entité-mère) en relation**

- Toujours applicable
- Les propriétés des sous-types deviennent des attributs optionnels (valeur non définie) dans la relation dérivée du sur-type
- Programmation complexe (e.g., requêtes concernant un sous-type qui interrogent la relation dérivée du sur-type)

# Transformation des spécialisations/généralisations

**Première solution : transformer le sur-type (entité-mère) en relation**

- Toujours applicable
- Les propriétés des sous-types deviennent des attributs optionnels (valeur non définie) dans la relation dérivée du sur-type
- Programmation complexe (e.g., requêtes concernant un sous-type qui interrogent la relation dérivée du sur-type)




EntitéA(idA, prop1A, prop2A, prop1B, prop2B, prop1C)

# Transformation des spécialisations/généralisations (2)

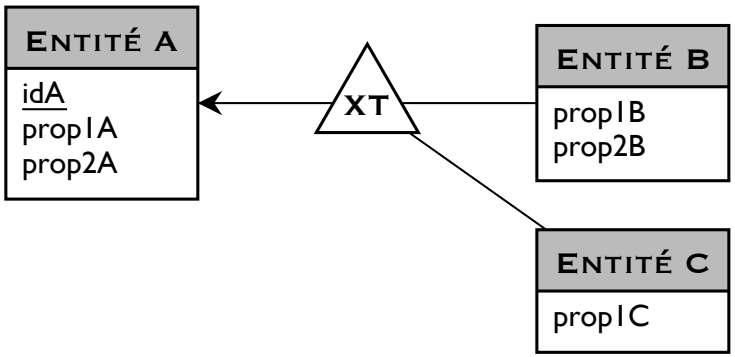
**Deuxième solution : transformer les sous-types (entités-filles) en relations**

- Applicable pour des généralisations totales et exclusives
- Redondances (les attributs du sur-type sont placés dans chaque relation dérivée d'un sous-type)
- Programmation complexe (e.g., requêtes concernant le sur-type qui interrogent chaque relation dérivé d'un sous-type)

# Transformation des spécialisations/généralisations (2)

**Deuxième solution : transformer les sous-types (entités-filles) en relations**

- Applicable pour des généralisations totales et exclusives
- Redondances (les attributs du sur-type sont placés dans chaque relation dérivée d'un sous-type)
- Programmation complexe (e.g., requêtes concernant le sur-type qui interrogent chaque relation dérivé d'un sous-type)




EntitéB(idA, prop1A, prop2A, prop1B, prop2B)

EntitéC(idA, prop1A, prop2A, prop1C)

# Transformation des spécialisations/généralisations (3)

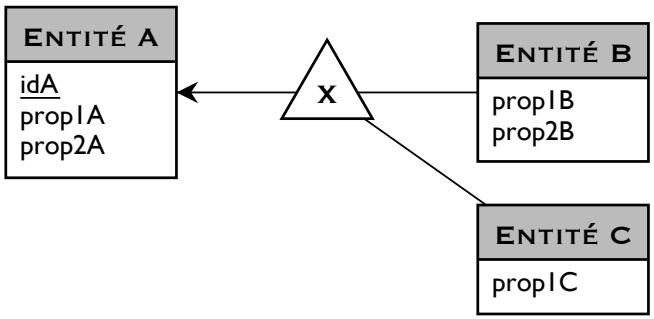
**Troisième solution : tout transformer en relations ordinaires**

- Les relations dérivées des sous-types récupèrent en clé étrangère les identifiants du sur-type
- Perte de sémantique, besoin de jointures
- Possible d'avoir une même valeur pour les clés primaires des relations sur-type/sous-types, mais gestion plus complexe

# Transformation des spécialisations/généralisations (3)

**Troisième solution : tout transformer en relations ordinaires**

- Les relations dérivées des sous-types récupèrent en clé étrangère les identifiants du sur-type
- Perte de sémantique, besoin de jointures
- Possible d'avoir une même valeur pour les clés primaires des relations sur-type/sous-types, mais gestion plus complexe



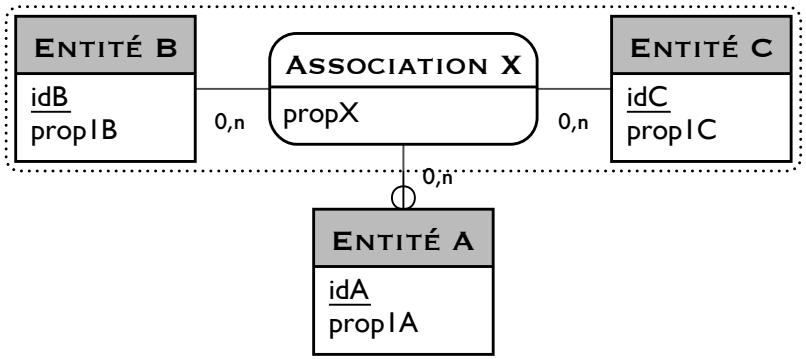

EntitéA(idA, prop1A, prop2A)
EntitéB(idB, prop1B, prop2B, #idA)
EntitéC(idC, prop1C, #idA)

# Transformation des agrégations

- **L'association agrégat devient une relation**
- Sa clé primaire se compose des identifiants de ses entités porteuses (dont celles de l'agrégat)
- Les propriétés de l'association deviennent des attributs dans sa relation

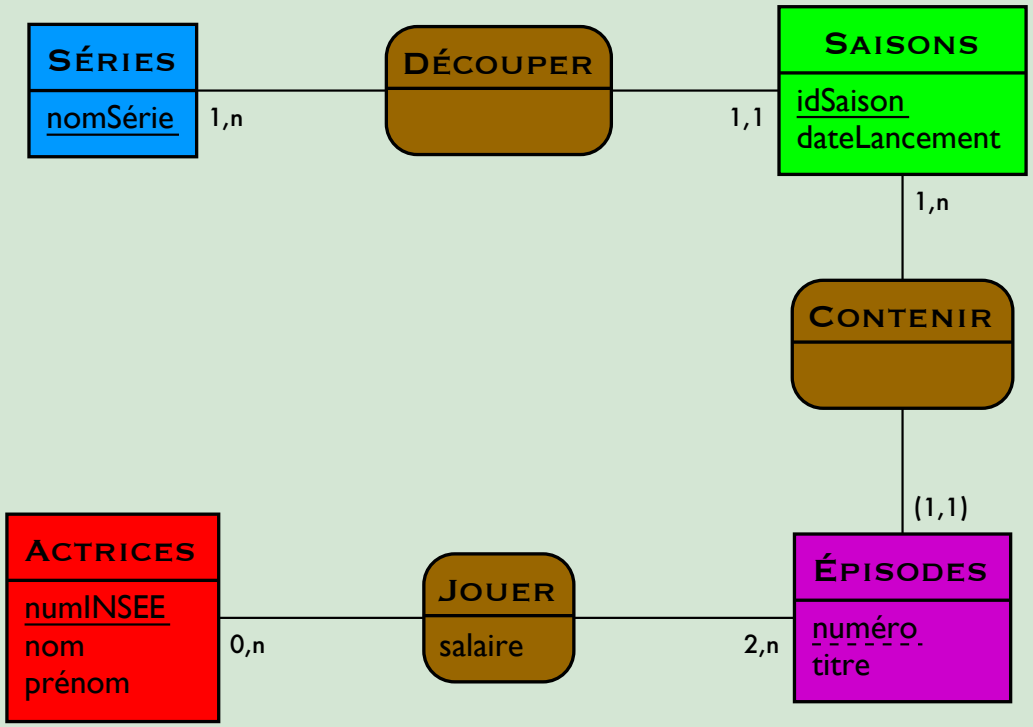
# Transformation des agrégations

- **L'association agrégat devient une relation**
- Sa clé primaire se compose des identifiants de ses entités porteuses (dont celles de l'agrégat)
- Les propriétés de l'association deviennent des attributs dans sa relation

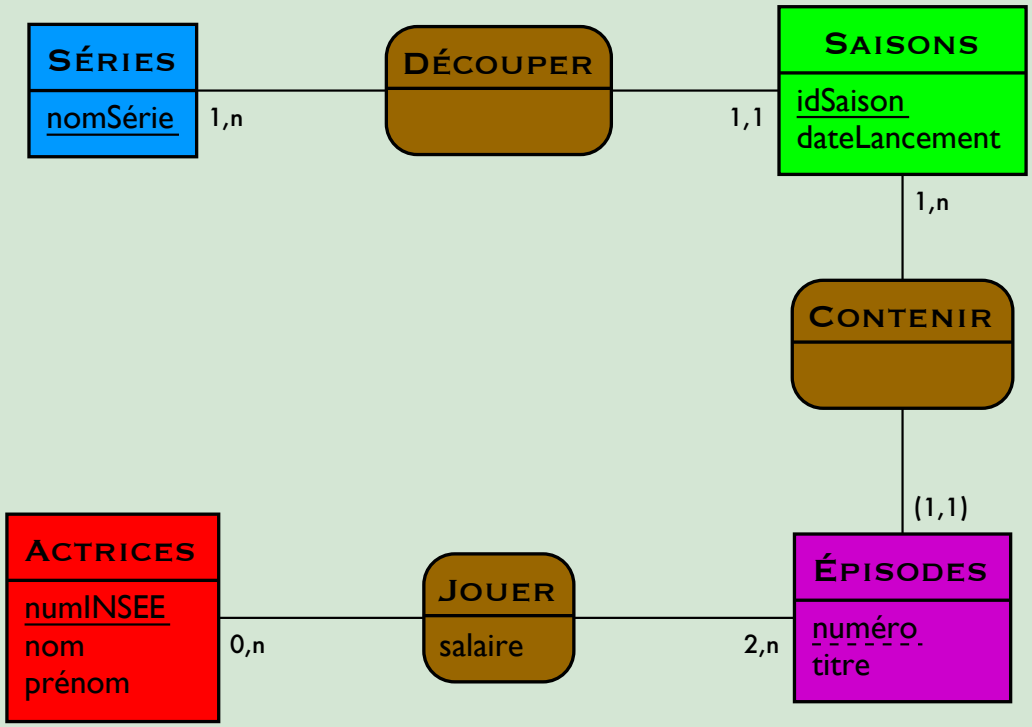



EntitéA(idA, prop1A)
EntitéB(idB, prop1B)
EntitéC(idC, prop1C)
AssociationX(#idA, #idB, #idC, propX)

# Exercice - transformation E/A vers relationnel



# Exercice - transformation E/A vers relationnel




SÉRIES (nomSérie)
SAISONS (idSaison, dateLancement, #nomSérie)
ÉPISODES (numéro, #idSaison, titre)
ACTRICES (numINSEE, nom, prénom)
JOUER (#numéro, #idSaison, #numINSEE, salaire)

# En résumé

Le contexte (importance des concepts, nature des données, contraintes du SGBD, fréquence des requêtes, etc.) permet de choisir une solution adaptée pour transformer les concepts du diagramme E/A étendu

Quelques exceptions :

- Une entité qui ne contient qu'une seule propriété de datation pour une association n-aire
- Une association avec cardinalité 0,1 ou 1,1 et qui inclut une ou plusieurs propriétés
- Une association avec des cardinalités 0,1 ou 1,1 de chaque côté

# Plan

# Définition

Normalisation : vérifier et améliorer un schéma relationnel afin de limiter les redondances et les mauvaises performances

Il existe huit formes normales :

- Basées sur les dépendances fonctionnelles
- En pratique, le contexte impacte sur le nombre de formes normales à utiliser (e.g., nombre de lectures/écritures)

**Dans cet enseignement, 3 premières formes normales**

---

http://fr.wikipedia.org/wiki/Forme_normale_(bases_de_donn%C3%A9es_relationnelles)

E. F. Codd. *Further Normalization of the Data Base Relational Model*. Data Base Systems : Courant Computer Science Symposia (1971)

# Dépendance fonctionnelle

Dans une relation, soient un ensemble d'attributs A et un ensemble d'attributs B

A détermine B (noté A→ B) si pour toute paire de n-uplets ayant les mêmes valeurs sur leurs attributs A, alors ces n-uplets ont les mêmes valeurs sur leurs attributs B

idE→ nomE
idE→ moyenneLycée
(idE, nomU, département)→ décision

https://fr.wikipedia.org/wiki/D%C3%A9pendance_fonctionnelle

# 1FN – Première forme normale

Une relation est en 1FN si :

- Les valeurs de ses attributs sont atomiques (non multi-valuées)
- Les valeurs de ses attributs sont scalaires (une seule donnée dépendant de la clé, e.g., album = *"Nirvana - Nevermind"*)
- Les valeurs des attributs sont constantes dans le temps

| nom | espèce | nourriture | âge | zoo | ville |
|---|---|---|---|---|---|
| Pam | paon | graines | 2 | zoo de Lyon | Lyon |
| Tigrou | tigre | viande | 5 | zoo de Lyon | Lyon |
| Léonie | lion | viande | 3 | parc Hérouval | Gisors |
| Léonie | loup | viande, graines | 6 | safari de Peaugres | Peaugres |

*L'attribut nourriture a des valeurs composites, l'attribut âge varie dans le temps*

# 2FN – Deuxième forme normale

Une relation est en 2FN si :

- Elle est en 1FN
- Chaque attribut non-clé dépend de l'intégralité de la clé primaire

| nom | espèce | nourriture | âge | zoo | ville |
|---|---|---|---|---|---|
| Pam | paon | graines | 2 | zoo de Lyon | Lyon |
| Tigrou | tigre | viande | 5 | zoo de Lyon | Lyon |
| Léonie | lion | viande | 3 | parc Hérouval | Gisors |
| Léonie | loup | viande, graines | 6 | safari de Peaugres | Peaugres |

*L'attribut nourriture dépend uniquement d'une partie de la clé (espèce)*

# 3FN – Troisième forme normale

Une relation est en 3FN si :

- Elle est en 2FN
- Un attribut non-clé ne dépend pas d'un ou plusieurs attributs non-clés

| nom | espèce | nourriture | âge | zoo | ville |
|---|---|---|---|---|---|
| Pam | paon | graines | 2 | zoo de Lyon | Lyon |
| Tigrou | tigre | viande | 5 | zoo de Lyon | Lyon |
| Léonie | lion | viande | 3 | parc Hérouval | Gisors |
| Léonie | loup | viande, graines | 6 | safari de Peaugres | Peaugres |

*L'attribut ville dépend de l'attribut zoo (qui n'est pas clé primaire)*

# Un moment de réflexion

Élève (idE, *nomE*, *moyenneLycée*, *effectifLycée*)
Candidature (#*idE*, #*nomU*, département, *décision*)
Université (nomU, *ville*, *effectif*)

| idE | nomE | moyenneLycée | effectifLycée |
|---|---|---|---|
| 123 | Ana | 19.5 | 1000 |
| 234 | Bob | 18 | 1500 |
| 345 | Chloé | 17.5 | 500 |
| 456 | Damien | 19.5 | 1000 |
| 543 | Chloé | 17 | 2000 |
| 567 | Éléonore | 14.5 | 2000 |
| 654 | Ana | 19.5 | 1000 |
| 678 | Farid | 19 | 200 |
| 765 | Joana | 14.5 | 1500 |
| 789 | Gisèle | 17 | 800 |
| 876 | Irène | 19.5 | 400 |
| 898 | Hector | 18.5 | 800 |

Table Élève

| nomU | ville | effectif |
|---|---|---|
| INSA | Lyon | 36000 |
| UCB | Lyon | 15000 |
| UJF | Grenoble | 10000 |
| UJM | Saint-Étienne | 21000 |

Table Université

| idE | nomU | département | décision |
|---|---|---|---|
| 123 | INSA | informatique | O |
| 123 | UCB | électronique | N |
| 123 | UCB | informatique | O |
| 123 | UJM | électronique | O |
| 234 | INSA | biologie | N |
| 345 | UJF | bioinformatique | O |
| 345 | UJM | bioinformatique | N |
| 345 | UJM | électronique | N |
| 345 | UJM | informatique | O |
| 543 | UJF | informatique | N |
| 678 | UCB | histoire | O |
| 765 | UCB | histoire | O |
| 765 | UJM | histoire | N |
| 765 | UJM | psychologie | O |
| 876 | UCB | informatique | N |
| 876 | UJF | biologie | O |
| 876 | UJF | biologie marine | N |
| 898 | INSA | informatique | O |
| 898 | UCB | informatique | O |

Table Candidature

Le jeu de données universitaire de Stanford vous semble-t-il bien modélisé (notamment *département* et *effectifLycée*) ?

# Travail personnel

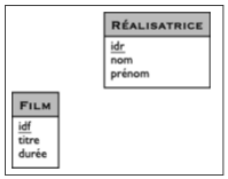
Suivre le tutoriel Mocodo (disponible sur la page de l'UE) pour préparer le TD et TP (~15 minutes)

**1.** Mocodo est un outil de modélisation, qui a l'avantage d'agencer automatiquement un diagramme entité/association. Il est utilisable directement en ligne :
https://www.mocodo.net/
La grande zone de texte permet de définir un diagramme, et le `bouton ventilateur` produit le dessin.
Dans l'onglet `Options`, activez la transformation en schéma relationnel et en SQL (voir capture à droite).



**2.** Voyons la syntaxe Mocodo pour définir les éléments d'un diagramme.
Une **entité** se déclare d'abord par son nom, puis un deux-point et une liste de propriétés (dont la première sert d'identifiant).
Copiez les entités `Film` et `Réalisatrice`, puis cliquez sur le `bouton ventilateur`.

```
Film : idf, titre, durée
Réalisatrice : idr, nom, prénom
```




**3.** Pour définir une **association**, on indique son nom puis une liste de cardinalité (2 caractères) et nom d'entité. Ajoutez l'association *Réaliser*, qui fait le lien entre un film et sa réalisatrice.
Appuyez sur la `tablette de chocolat` pour tester d'autres agencements de votre diagramme.

```
Réaliser, 11 Film, On Réalisatrice
```