

# Base de données

---

## Chapitre 4 : Formalisation des bases de données relationnelles

Stéphane Malandain – sbd – 2024

# Introduction

---

# introduction

---

## Définition :

Une base de données est une collection d'informations persistantes organisée de manière à pouvoir être facilement manipulée

Elle est définie par :

- Des entités
- La nature des liens entre ses entités
- Des contraintes

# Base de données relationnelles

---

## Base de données relationnelle

Une base de données relationnelle permet de regrouper des données structurées dans un ensemble de **tables** organisées en **lignes** et en **colonnes**. Elle permet de renforcer les contraintes d'intégrité.

# Formalisation

---

# Le modèle relationnel

---

**Réalisé par Edgar F. Codd en 1970**

Algèbre relationnel

Basé sur des fondements mathématiques

- Théorie des ensembles
- concepts de relations
- logique du premier ordre

Objectifs :

- comment décrire formellement une base de données ?

# Le modèle relationnel

---

## Utilité

Représenter de manière non-ambiguë un moyen de décrire les données, leur structure et les opérations sans considération informatique.

Bénéficier de l'apport des mathématiques (pour démontrer et prouver...)

Permet une totale indépendance entre les **programmes** (impératifs et procéduraux) et l'**organisation des données** (déclarative/descriptive).

# Ensemble

---

## Ensemble

Un ensemble est une collection d'éléments. Chaque élément est **unique**.

# Produit cartésien

---

## Produit cartésien

Le produit cartésien de deux ensembles  $X$  et  $Y$ , noté  $X \times Y$ , est l'ensemble des couples  $(x, y)$  tel que  $x \in X, y \in Y$ .

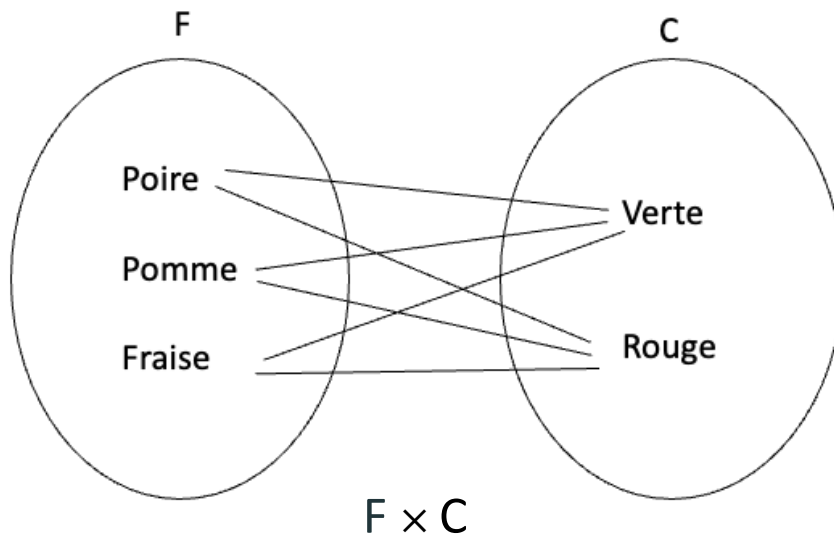
Il est noté également  $\{(x, y) \mid x \in X, y \in Y\}$

# Produit cartésien

## Exemple

$F = \{ \text{Poire, Pomme, Fraise} \}$      $C = \{ \text{Verte, Rouge} \}$

$F \times C = \{ (\text{Poire, Verte}), (\text{Poire, Rouge}), (\text{Pomme, Verte}), (\text{Pomme, Rouge}), (\text{Fraise, Verte}), (\text{Fraise, Rouge}) \}$



*fig. 1 : représentation ensembliste*

**F X N**

**Fruit (F)    Couleur (C)**

Poire	Verte
Poire	Rouge
Pomme	Verte
Pomme	Rouge
Fraise	Verte
Fraise	Rouge

*fig. 2 : rep. tabulaire*

# Schéma d'une relation ( = table )

---

## Schéma d'une relation

Le schéma d'une relation R dénoté  $R(A_1, A_2, \dots, A_n)$  est composé d'un nom R et d'attributs  $A_1, A_2, \dots, A_n$

Une table est le nom commun d'un schéma de relation.

Exemple :

`Visiteur(id_visiteur, prenom, nom, email)`

`Conference(id_conf, intitule, date_debut, date_fin, prix)`

`Hotel(id_hotel, nom_hotel, adresse)`

# Relation

---

## Relation

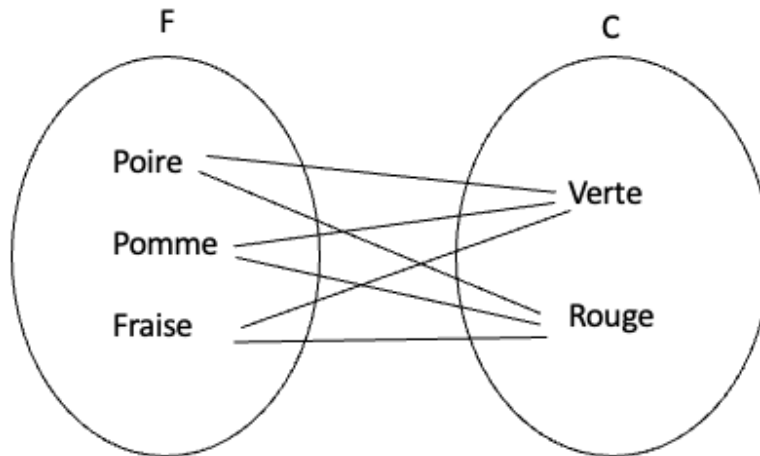
Une relation  $r$  d'un schéma  $R$ , notée  $r(R)$ , est un sous-ensemble du produit cartésien défini par  $R : r(R) \subseteq A_1 \times A_2 \times \dots \times A_n$ .

Une relation est un ensemble de tuples (*n-uplets*)  
 $(a_1, a_2, \dots, a_n), a_i \in A_i$

Un tuple représente un enregistrement (une ligne) d'une table alors qu'une relation représente l'ensemble des enregistrements d'une table.

# Relation

## Exemple



**F x C**  
**Fruit (F)    Couleur (C)**

Poire	Verte
Poire	Rouge
Pomme	Verte
Pomme	Rouge
Fraise	Verte
Fraise	Rouge

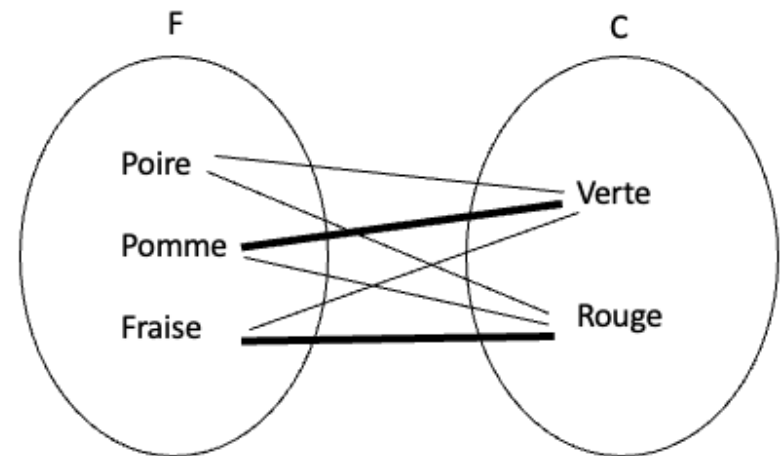


schéma: Panier(fruit, couleur)  
relation: panier(Panier)  $\subseteq$  F x C

**Fruit (F)    Couleur (C)**

<del>Poire</del>	<del>Verte</del>
<del>Poire</del>	<del>Rouge</del>
Pomme	Verte
<del>Pomme</del>	<del>Rouge</del>
<del>Fraise</del>	<del>Verte</del>
Fraise	Rouge

# Relation

---

Par définition, une relation est également un ensemble

**Panier**

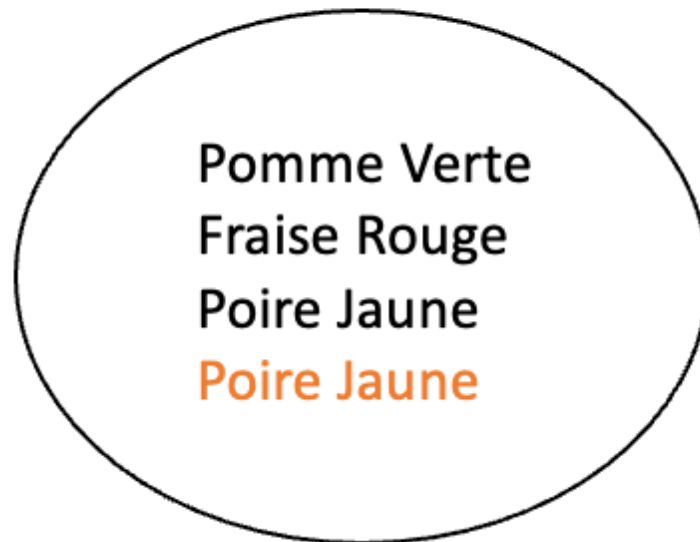


## Relation et tuple

---

Par définition, un ensemble est une collection d'éléments uniques. Il n'existe donc **pas deux tuples de mêmes valeurs** dans une relation.

Panier



# Clé primaire

---

## Clé primaire

La clé primaire d'un schéma est un sous-ensemble d'attributs du schéma de la relation permettant d'identifier de manière unique un tuple.

- Deux tuples distincts d'une relation ne peuvent avoir la même clé (contrainte d'unicité).
- Une clé est obligatoirement renseignée
- Il peut exister plusieurs clés candidates
- Une clé artificielle peut être créée
  - si aucune clé n'est candidate
  - *(si les clés candidates sont mal adaptées pour l'indexation)*

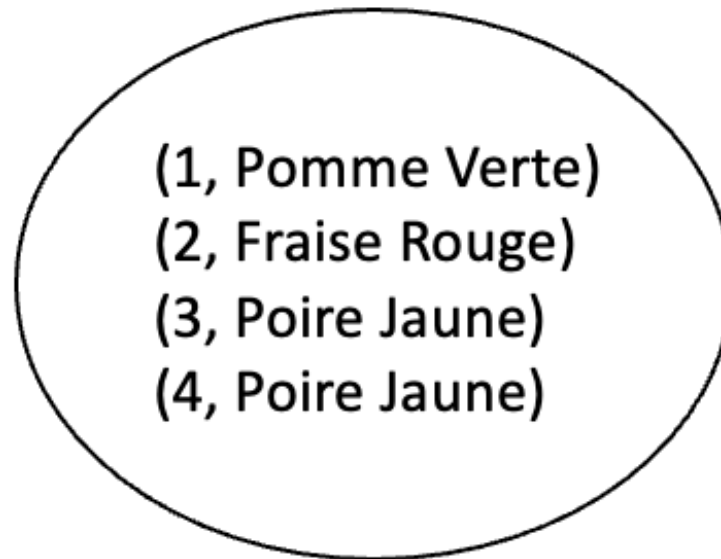
## Clé primaire

---

### Exemple

`Panier(id_panier, fruit, couleur)`

Panier



## Clé primaire

---

La clé primaire primaire décrit une contrainte forte à elle seule. Dans l'exemple précédent, elle précise :

- Connaissant un identifiant, celui-ci ne peut correspondre qu'à un fruit et une couleur.

# Clé étrangère

---

## Clé étrangère

Une clé étrangère est un attribut (ou un ensemble d'attributs) référençant une clé primaire d'une autre table.

La valeur doit exister dans l'autre relation (contrainte d'intégrité référentielle ou **dépendance d'inclusion**).

- Elle permet d'éviter les redondances et anomalies
- Elle indique une cardinalité (correspond à un)

## Clé étrangère

### Redondance et cardinalité

id_conf	nom	id_theme	intitulé
1	ScalaDays 2018	1	MicroService
2	LambdaDays 2019	2	Prog. fonctionnelle
3	Curry On 2019	3	Système formel
4	ScalaDays 2020	2	Prog. fonctionnelle

*fig. 3: relation avec des redondances*

id_conf	nom	id_theme	id_theme	intitulé
1	ScalaDays 2018	1	1	MicroService
2	LambdaDays 2019	2	2	Prog. fonctionnelle
3	Curry On 2019	3	3	Système formel
4	ScalaDays 2020	2		

*fig. 4 : séparation + clé étrangère*

# Clé étrangère

---

## Formalisation

```
Theme(id_theme, intitule)  
Conference(id_conference, nom, id_theme)  
id_theme  $\subseteq$  Theme.id_theme
```

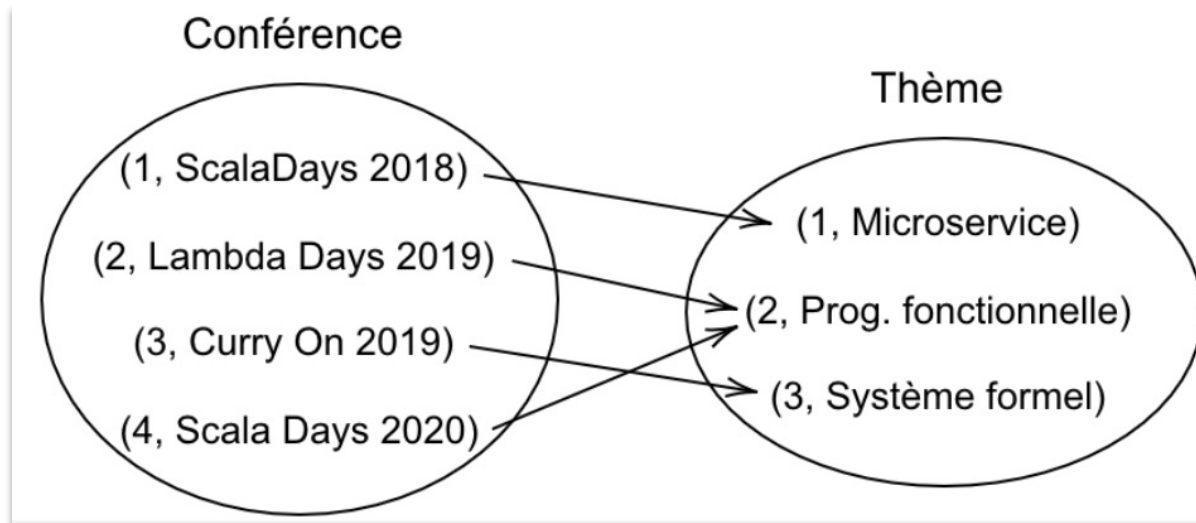
Il est utile de se représenter cette "liaison" à l'aide d'un diagramme sagital et de flèches

## Clé étrangère

Représentons les associations à l'aide d'arcs (flèches). Le sens a une importance, il décrit un lien de déduction entre l'origine (conférence) et son extrémité (thème).

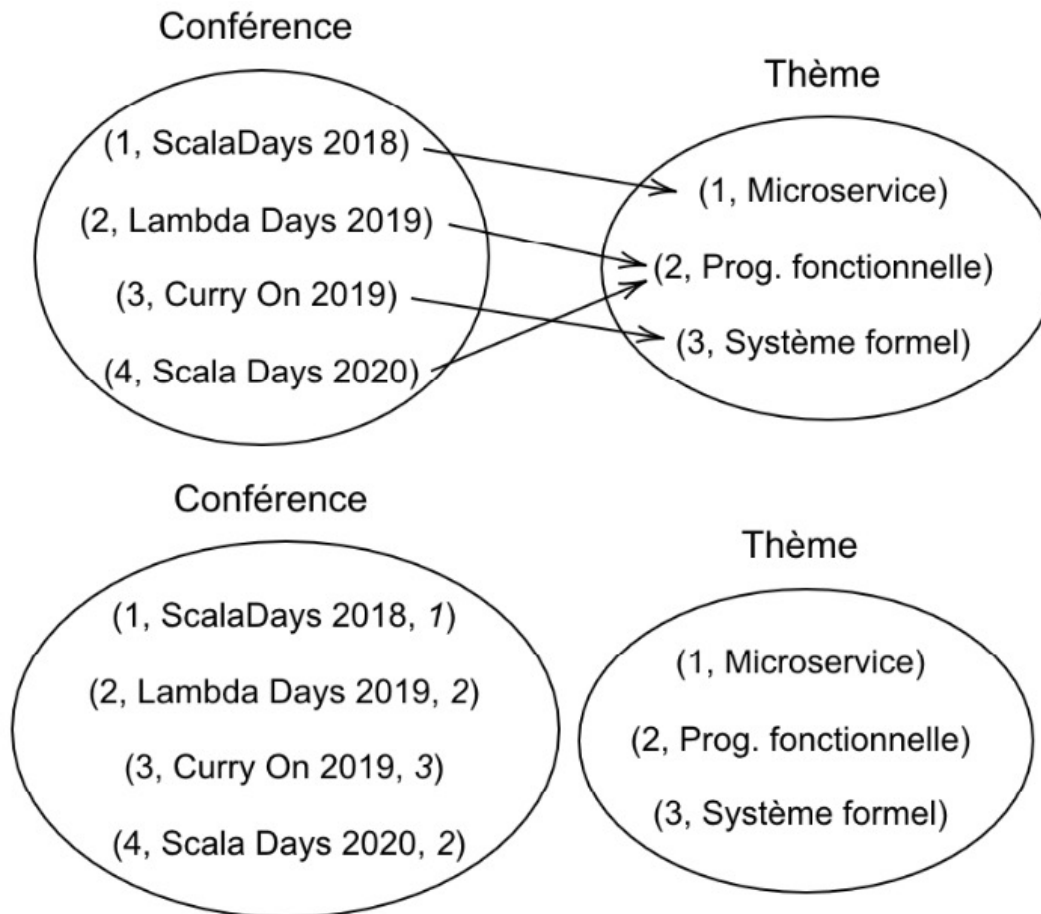
### Interprétation de l'illustration

- connaissant une conférence, je peux connaître son thème; une conférence n'a donc qu'un seul thème (card. max de 1)
- étant donné qu'une conférence n'a qu'un thème, chaque conférence devient l'**origine** d'un arc.



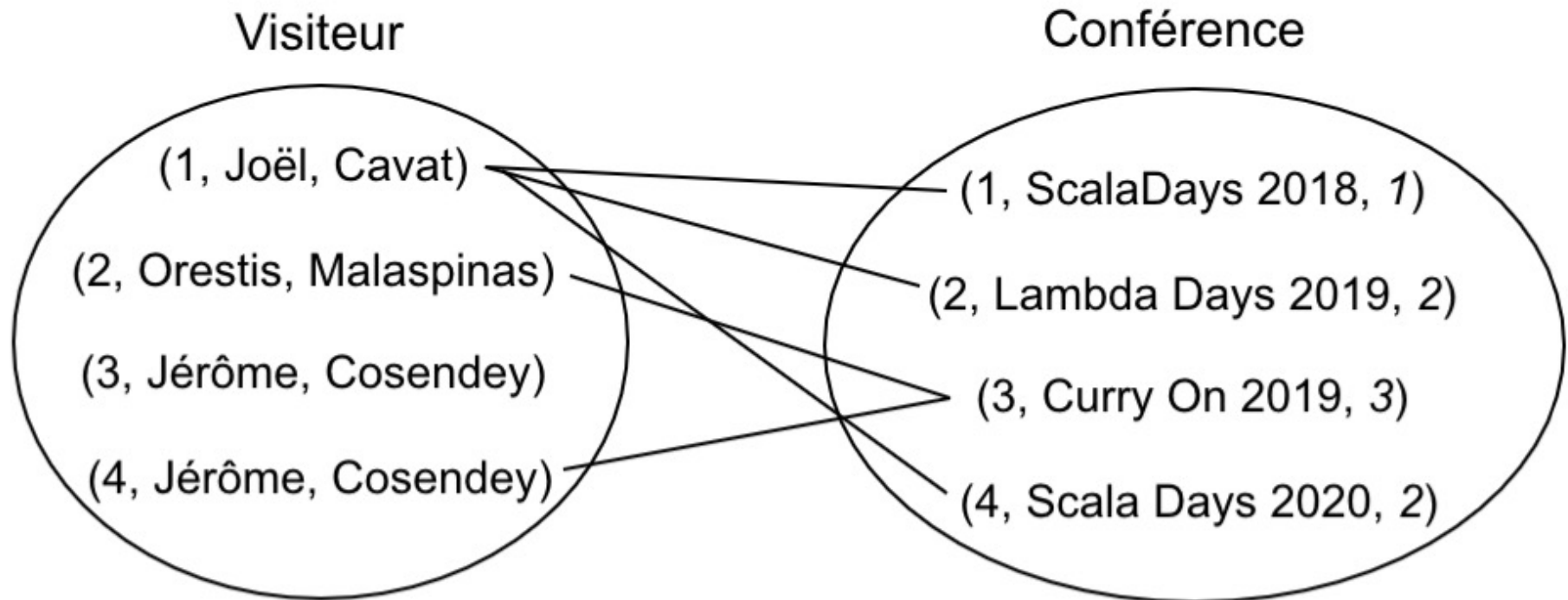
## Clé étrangère

Dans l'univers relationnel, les arcs n'existent pas. En les supprimant, nous comprenons l'utilité de la clé étrangère : elle permet de représenter une association. Elle apparaît du côté de l'origine de l'arc.



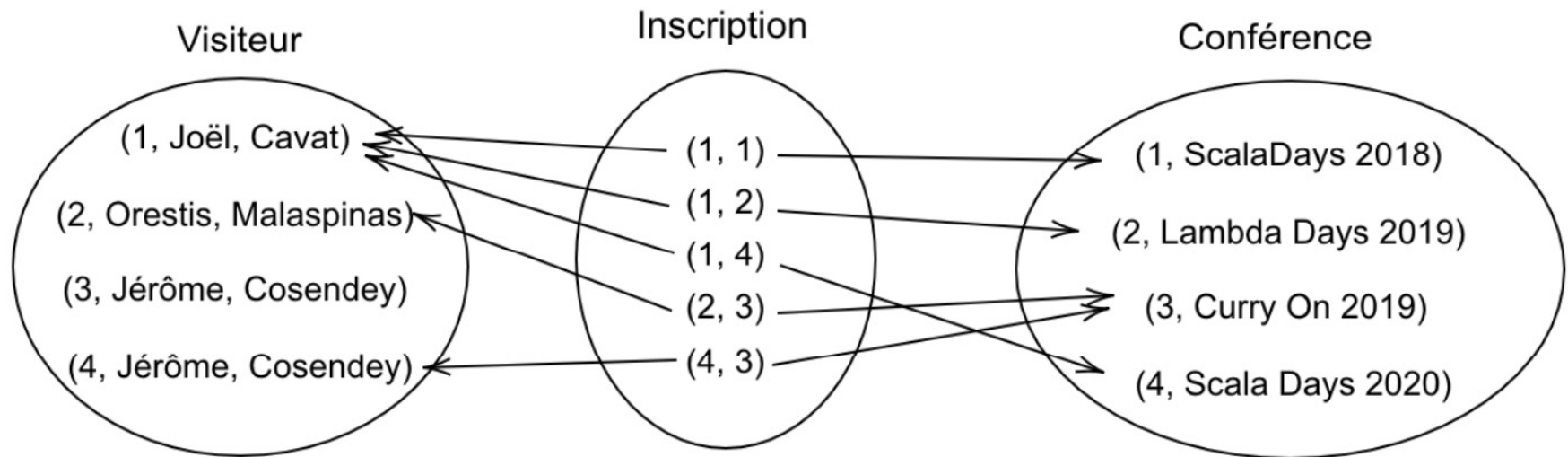
## Clé étrangère

Association plusieurs-à-plusieurs ???



## Clé étrangère

Association plusieurs-à-plusieurs ???



# Fonction et dépendance fonctionnelle

---

## Fonction

Une fonction d'un ensemble A vers un ensemble B est une relation (un sous ensemble  $A \times B$ ) où chaque élément de A est unique (appartient à un et un seul couple de la relation)

## Dépendance fonctionnelle

Les dépendances fonctionnelles sont des contraintes qui décrivent les liens qui peuvent exister entre des attributs.

## Dépendance fonctionnelle

---

Une DF, dénotée  $X \rightarrow Y$ , entre deux ensembles d'attributs X et Y d'une relation R, spécifie une contrainte entre ces deux ensembles.

$X \rightarrow Y$  peut être interprété de différentes manières :

- Connaissant X je peux déduire Y
- Connaissant X il ne peut y avoir qu'un Y
- X détermine Y

On dit aussi que X détermine Y (ou que Y dépend fonctionnellement de X), si et seulement s'il existe une fonction qui à partir de toute valeur de X détermine une valeur unique de Y.

Remarques :

- X est appelé le **déterminant**
- Y est appelé le **déterminé**

# Dépendance fonctionnelle

---

Exemple 1 :

1.  $\{\text{id\_theme}\} \rightarrow \{\text{intitule}\}$
2.  $\{\text{id\_conference}\} \rightarrow \{\text{nom, id\_theme}\}$

Signification:

1. Connaissant l'identifiant d'un thème, il est possible de connaître son intitulé. Sémantiquement, correspond à un thème
2. L'identifiant d'une conférence nous permet de connaître le nom de la conférence et l'identifiant de son thème associé. Sémantiquement, correspond à une conférence.

# Dépendance fonctionnelle

---

Exemple 2 :

1. {no\_isbn} → {titre, description}
2. {no\_exemplaire} → {no\_isbn, date\_acquisition, prix}
3. {no\_exemplaire, no\_emprunteur, date\_emprunt} → {date\_retour, date\_echeance}

Quizz : écrivez leur signification et trouvez une sémantique

1. ...
2. ...
3. ...

## Dépendance fonctionnelle

---

### Exemple 2 :

1. {no\_isbn} → {titre, description}
2. {no\_exemplaire} → {no\_isbn, date\_acquisition, prix}
3. {no\_exemplaire, no\_emprunteur, date\_emprunt} → {date\_retour, date\_echeance}

Quiz : écrivez leur signification et trouvez une sémantique

1. Connaissant le numéro isbn d'un livre, il est possible de connaître son titre et sa description. Sémantiquement, correspond à un livre
2. Connaissant le numéro d'exemplaire d'un livre, il est possible de connaître son isbn, sa date d'acquisition et son prix. Sémantiquement, correspond à un exemplaire acheté à la bibliothèque
3. Connaissant le n° d'exemplaire, le n° d'emprunteur et la date de l'emprunt, il est possible de connaître la date de retour et l'échéance de celle-ci. Sémantiquement, correspond à un emprunt

# Dépendance fonctionnelle

---

## Utilité :

En plus de permettre de modéliser des contraintes supplémentaires, elles permettent de déterminer les identifiants.

## Remarque :

Les attributs déterminants sont généralement **une clé primaire ou unique**

# Dépendance fonctionnelle

## Exemple 1

Le schéma `Personne(no_avs, nom, prenom, date_naissance)` et sa  
DF  $\{no\_avs\} \rightarrow \{nom, prenom, date\_naissance\}$  notée :

```
Personne(no_avs, nom, prenom, date_naissance)  
DF1: {no_avs} → {nom, prenom, date_naissance}
```

se simplifie ainsi :

```
Personne(no_avs, nom, prenom, date_naissance)
```

Le choix de la clé décrit une DF.

# Dépendance fonctionnelle

## Exemple 2

Le schéma `Client(no_client, nom, prenom, no_tel)` et ses DF :

$\{no\_client\} \rightarrow \{nom, prenom\}$

$\{no\_tel\} \rightarrow \{nom, prenom\}$

se simplifie ainsi :

```
Client(no_client, nom, prenom, no_tel)
```

ou de préférence (préférez une seule clé primaire)

```
Client(no_client, nom, prenom, no_tel)  
No_tel UNIQUE
```

# Dépendance fonctionnelle

---

## Quiz

Quelle serait la limitation de cette DF ? Est-elle plus restrictive ou plus permissive ?

$\{\text{no\_exemplaire}, \text{no\_emprunteur}\} \rightarrow \{\text{date\_emprunt}, \text{date\_retour}, \text{date\_echeance}\}$

par rapport à celle-ci :

$\{\text{no\_exemplaire}, \text{no\_emprunteur}, \text{date\_emprunt}\} \rightarrow \{\text{date\_retour}, \text{date\_echeance}\}$

# Schéma de base de données relationnelle

---

## Schéma de base de données relationnelle

Le schéma d'une base de données relationnelle est composé de l'ensemble des schémas des relations et de l'ensemble des dépendances fonctionnelles.

## Schéma de base de données relationnelle

Modèle EA	Modèle relationnel (formel).	Modèle relationnel (courant)
Attribut	Attribut	Attribut
Identifiant	Clé primaire	Clé primaire
Type d'entité	Schéma de relation	Table
Entité	Tuple	Enregistrement
-	Relation	Ensemble des enregistrements
Association	<i>représenté par CI réf.</i>	<i>représenté par une clé étrangère</i>

## Dépendance d'inclusion

---

Nous avons vu qu'une clé étrangère d'un schéma de relations doit référencer la clé primaire d'un autre schéma de relations.

La valeur de la clé étrangère doit donc exister dans l'autre relation.

La notation que nous avons employé correspond en fait à une **dépendance d'inclusion (DI)**.

```
Theme(id_theme, intitule)  
Conference(id_conference, nom, id_theme)  
id_theme  $\subseteq$  Theme.id_theme
```

## Dépendance d'inclusion

---

De manière générale :

- Une **DI** indique que les valeurs d'un ou plusieurs attributs d'un enregistrement doivent exister dans une autre relation
  - 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 : cas d'utilisation

---

Imaginons une plateforme en ligne de type Amazon qui met en relation des fournisseurs et des clients.

Les slides suivantes proposent deux schémas de relations qui permettent une telle modélisation.

## Dépendance d'inclusion : cas d'utilisation

- `Livraison(...)` : ceci permet de savoir les produits livrés par un fournisseur. Un même produit peut être livré par plusieurs fournisseurs différents.
- `Achat(...)` : un client peut acheter un produit auprès du fournisseur de son choix.

extrait du schéma de base de données:

`Achat(id_achat, id_fourn, id_client, id_produit, date)`

`id_fourn  $\subseteq$  Fournisseur.id_fournisseur`

`id_client  $\subseteq$  Client.id_client`

`id_produit  $\subseteq$  Produit.id_produit`

`Livraison(id_fournisseur, id_produit)`

`id_produit  $\subseteq$  Produit.id_produit`

`id_fournisseur  $\subseteq$  Fournisseur.id_fournisseur`

## Dépendance d'inclusion : cas d'utilisation

Achat(id\_achat, id\_fourn, id\_client, id\_produit, date), ref: ...

Livraison(id\_fournisseur, id\_produit), ref: ...

### livraison(Livraison)

id_fourn	id_produit
F1	P1
F1	P2
F2	P1
F3	P4

### achat(Achat)

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

Remarque :

F1 ne livre pourtant pas le produit P4

## Dépendance d'inclusion : cas d'utilisation

```

Achat(id_achat, id_fourn, id_client, id_produit, date)
  (id_fourn, id_produit) ⊆ Livraison.(id_fournisseur, id_produit)
  id_client ⊆ Client.id_client
Livraison(id_fournisseur, id_produit)
  id_produit ⊆ Produit.id_produit
  id_fournisseur ⊆ Fournisseur.id_fournisseur
  
```

### livraison(Livraison)

id_fourn	id_produit
F1	P1
F1	P2
F2	P1
F3	P4

### achat(Achat)

id_achat	id_fourn	id_produit	id_client	date
A1	F2	P1	C2	d1
A2	<del>F1</del>	<del>P4</del>	<del>C1</del>	<del>d2</del>

## Dépendance d'inclusion : cas d'utilisation

### Première version (sans contrainte)

Achat(id\_achat, id\_fourn, id\_client, id\_produit, date)

id\_fourn  $\subseteq$  Fournisseur.id\_fournisseur

id\_client  $\subseteq$  Client.id\_client

id\_produit  $\subseteq$  Produit.id\_produit

Livraison(id\_fournisseur, id\_produit)

id\_produit  $\subseteq$  Produit.id\_produit

id\_fournisseur  $\subseteq$  Fournisseur.id\_fournisseur

### Seconde version (améliorée)

Achat(id\_achat, id\_fourn, id\_client, id\_produit, date)

**(id\_fourn, id\_produit)  $\subseteq$  Livraison.(id\_fournisseur, id\_produit)**

id\_client  $\subseteq$  Client.id\_client

Livraison(id\_fournisseur, id\_produit)

id\_produit  $\subseteq$  Produit.id\_produit

id\_fournisseur  $\subseteq$  Fournisseur.id\_fournisseur